SIMULATION STUDY OF THE ROLL CONTROL LOOP OF AN OPERATIONAL SURFACE TO AIR MISSILE

David Parkin Hall

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

SIMULATION STUDY OF THE ROLL CONTROL LOOP OF AN OPERATIONAL SURFACE TO AIR MISSILE

by

David Parkin Hall

Thesis Advisor: G. J. Thaler

December 1973

Simulation Study of the Roll Control Loop of an Operational Surface to Air Missile

by

David Parkin Hall
Lieutenant, United States Navy
B.S.E.E., University of Michigan, 1966

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 1973

ABSTRACT

This thesis studies the roll loop of an operational surface to air missile as an example of a typical self adaptive control system. The missile studied also exhibits a phenomenum called ROLL WANDER, a low frequency oscillation in missile roll orientation. Because of this, the thesis is developed in three major areas of study. First, is the study of the missile roll control loop. Second, is a study of the phenomenon of ROLL WANDER, with the production of this phenomenon in the computer simulation. Third and finally is the implementation, in the computer simulation, of a self adaptive notch filter to suppress the occurence of ROLL WANDER.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

Figure

ACKNOWLEDGEMENT

This student wishes to express his profound thanks to Mr. Robert Hall and Mr. Chris McArdle of the Pomona Division of the General Dynamics Corporation, along with LCDR Donald G. McDougall, Naval Plant Representative's Office, Pomona for their invaluable assistance in the construction and testing of the missile model.

Appreciation is also gratefully extended to the staff of the U.S. Naval Postgraduate School Computer Facility; in particular Mrs. Lois M. Brunner and Mr. Richard E. Donat for their technical assistance in the production of DSLPLOT and Mr. Mannas Anderson, Miss Kristina Bertwell, Ms. Thelma M. Tegtmeier and Mr. Edwin V. Donnellan for their encouragement and patience during the many hours of computer time needed for the perfection and testing of this thesis.

Finally and most importantly this student wishes to thank Dr. George J. Thaler, Professor of Electrical Engineering, U.S. Naval Postgraduate School. It was through Dr. Thaler's guidance, persistence and understanding that this thesis was at all possible.

I. CONTROL SYSTEMS

A. INTRODUCTION

The objective of this thesis was to study a self-adaptive control system through the use of a computer simulation. The control system studied was the roll control loop of an operational surface to air missile.

This particular control loop uses a bistable actuator (a nonlinear element) as the primary mover of the loop. This method of control, a full throw "bang bang" system is commonly used in control systems. The control loop is driven either full positive or full negative, depending upon the polarity of the error signal. Since the system reacts with full response, the output is driven rapidly in opposition to the error and the error signal is eventually driven to the opposite polarity. The control then changes sign and the output is driven in the opposite direction. In this manner, the error signal is nulled out.

Bang bang control systems are in wide-spread use for such systems as surface to surface missiles, surface to air missiles, and some types of surface and submarine torpedoes. All of these systems require an accurate, reliable type one system producable at a minimum of expense. A full throw control system can provide these requirements.

In the case of a surface to air or air to air missile, however, there are added problems that the control system must be able to cope with. It must be able to operate at

various altitudes and at various speeds. The correct control response to an error signal of a given magnitude at high altitude (thin atmosphere) would be a drastic over response to the same error signal at low altitude (dense atmosphere). To correct for this, the control system must be self adaptive. It must respond to the error signal at a rate appropriate with its medium. Another constraint placed upon the control system is the requirement not to disturb the missile guidance and seeking/tracking mechanisms. While a full throw control system is inexpensive and reliable, it does introduce a constant oscillation in the position output. This can be disruptive to the other missile systems mentioned.

The surface to air guided missile studied in this thesis utilizes a bang bang type of control system which is both self-adaptive and non-disruptive to other missile functions.

B. DITHERING IN THE ROLL SYSTEM

Figure 1-1 is a block diagram of the roll control loop. The bistable element provides the control command to the tail control surface actuators. The use of this nonlinear element in the control loop causes a limit cycle oscillation to appear. It is this limit cycle oscillation which provides the self adaptive feature of the autopilot, and at the same time allows for smooth loop response while using a bang bang type control system. With no error or command signal present, the limit cycle oscillation provides a constant amplitude dither signal to the tail control surface servos. The dither

BLOCK DIAGRAM of the ROLL CONTROL LOOP

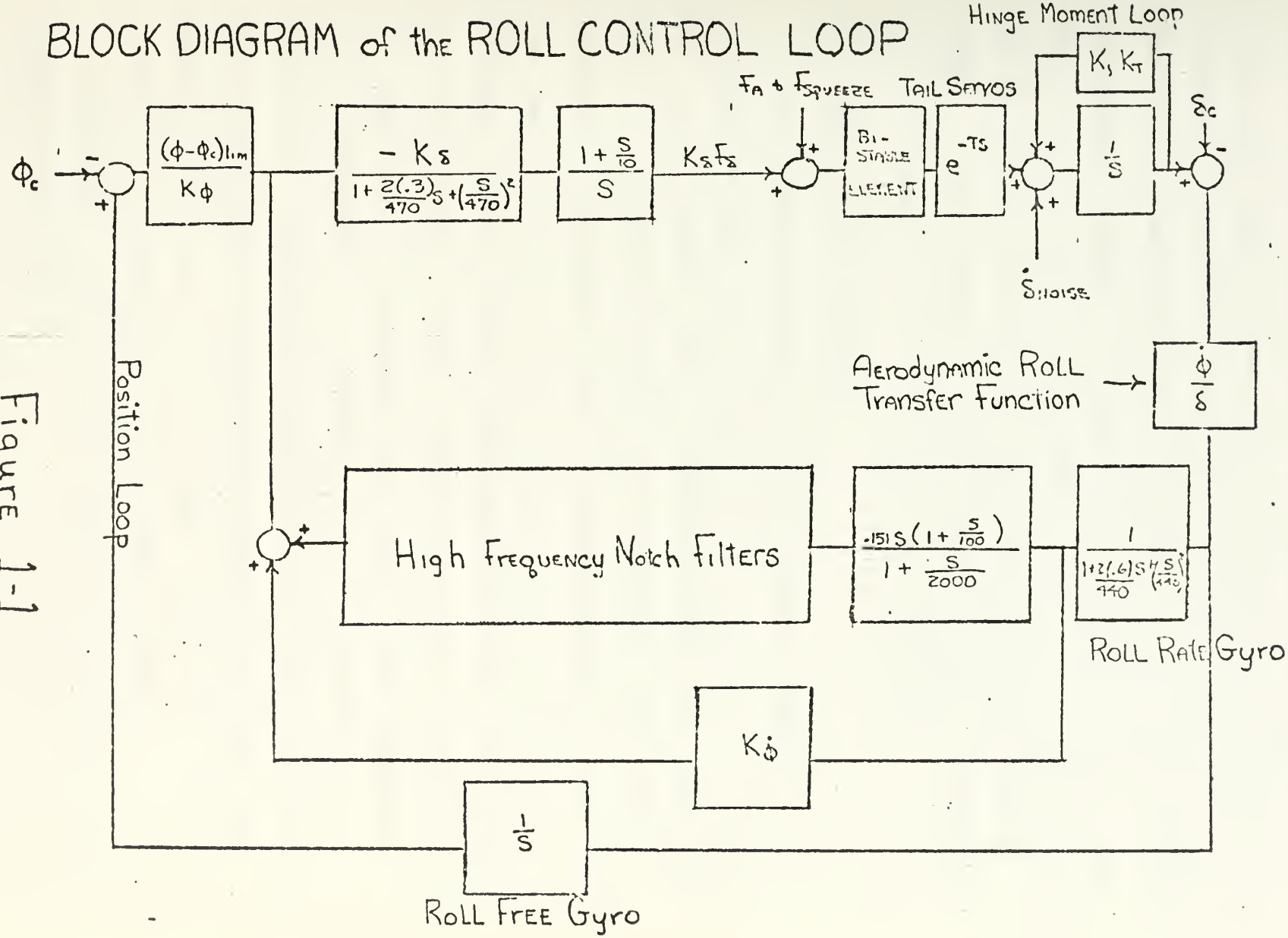


Figure 1-1

with zero error signal has equal negative and positive components; and as a result, the clockwise and counter clockwise roll torques produced by the tail surfaces cancel. The missile therefore dithers slightly in roll but maintains a constant average roll orientation. The amplitude of the dither signal, and thus the adaptive gain characteristics of the autopilot is a function of G, the roll moment coefficient. G is a measure of the aerodynamic gain, a property of the medium in which the missile is traveling

C. SELF-ADAPTIVE FEATURE

The autopilot derives its self-adaptive feature by varying the amplitude of limit cycle oscillation, or dither, in the roll control loop. A study of Figure 1-1 shows that there are three basic loops. The outer two loops provide position (ϕ') and velocity ($\dot{\phi}'$) feedback paths for stability and damping. The inner loop also provides some rate ($\ddot{\phi}'$) feedback, but of greater importance, the inner loop coupled with the bistable element produces the limit cycle oscillation. The roll loop of the missile was used for the dithering loop, since dithering in the other control loops would not be practical. Dithering in the pitch loop would be intolerable due to the amount of body motion sensed by the head rate gyros. Similarly, a dither in the squeeze mode would not provide the aerodynamic adaptive gain features.

Equation 1 is the dynamic equation for the rigid body motion of the missile.

$$\ddot{\phi} = G\delta + F\dot{\phi} \tag{1}$$

$$\therefore \quad \frac{\phi}{\delta} = \frac{G}{(s-F)} \tag{1a}$$

where: G = the aerodynamic moment

F = the aerodynamic damping

This transfer function, called the "aerodynamic roll transfer function" produces the inherent self-adaptive feature.

$$G = 1481 \lambda S d M^2 \frac{57.3}{I_R} C_{L\delta} \tag{1b}$$

where λ = static air pressure ratio

M = mach no.

I_R = roll inertia

$C_{L\delta}$ = roll damping coefficient

$$F = 1481 \lambda S d M^2 \frac{57.3}{I_R} \frac{b}{V} C_{L\left(\frac{bp}{V}\right)} \tag{1c}$$

where: V = velocity

S, d, b = aerodynamic constants

As can be seen by Equations (1b) and (1c), the gain of the roll transfer function is dependent on flight conditions. Therefore, as the missile's environment changes, so does the

gain of the roll transfer function and, accordingly, the amplitude of the position loop's limit cycle oscillation, or dither.

D. ROLL CONTROL

The controlling action of the self-adaptive loop is shown in Figure 1-2. The tail control surface drive motors are driven by the output of the bistable element at a constant rate in either the positive or negative direction. When an error signal is present, it appears as a dc bias signal summed with the roll loop's natural dither oscillation (Figure 1-2a). The algebraic sum of the two signals is fed to the bistable element (Figure 1-2b). As its name implies, the bistable element has only two states, full positive and full negative. When the value of $K_{\delta}F_{\delta}$ (the summation signal) passes through zero the bistable element shifts states. The action of the dc bias produced by the error signal is to cause the square wave output of the bistable element to remain positive or negative longer during a complete cycle, as appropriate (Figure (1-2c). Since the tail control surface motors act to integrate this output, the result is a dithering movement of the control surfaces modulated by the appropriate control surface movement to null the roll error signal.

As previously discussed, the gain of the roll transfer function determines the amplitude of the dither oscillation. As a missile flies higher, the gain decreases and therefore

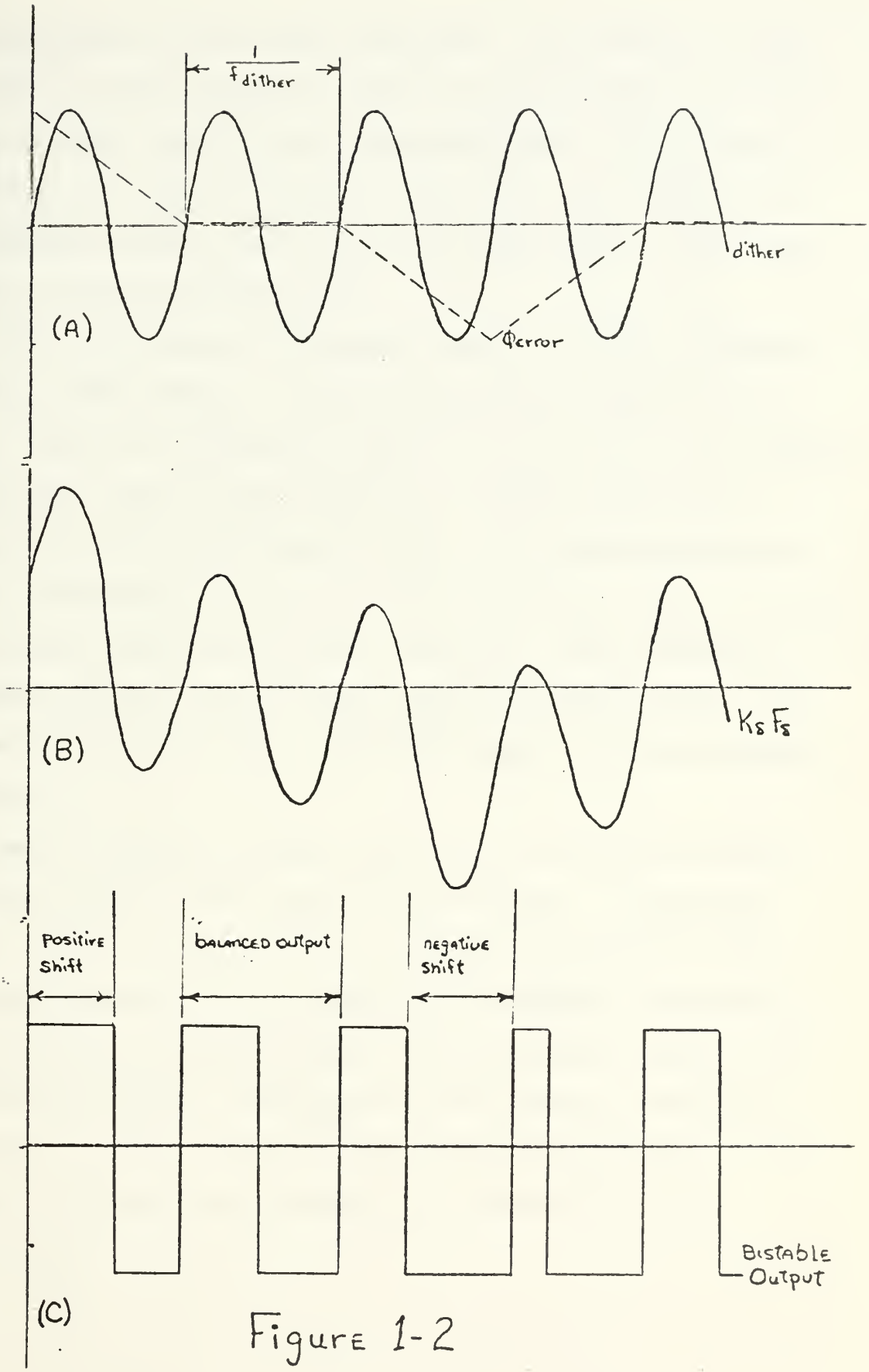

1/f dither
dither
(A)
φerror
(B)
Ks Fs
Positive shift
balanced output
negative shift
Bistable Output
(C)

Figure 1-2

the amplitude of the dither decreases. The effect of the dc bias is more pronounced with the result that the control loop is more sensitive and has higher gain. Conversely, at lower altitudes or higher speeds, the dither amplitude is greater and the error signal has less effect. The loop gain is reduced (Figure 1-3a).

The dither imposes a maximum signal strength allowable for the error signal. If at a given dither amplitude an error signal is encountered which is greater than the dither amplitude, then the oscillation is shifted completely to one side of the axis (Figure 1-3b). The bistable element stays continuously in one state and the tail control surfaces are driven continually in one direction. Loop response diverges and the loop becomes unstable (Figure 1-3c). To prevent this occurrence, the error signal is electronically limited.

Smooth response in the roll loop is obtained by superimposing the dc error signal on the dither. In fact the error signal is not a pure dc signal, but is also a time varying waveform. It is important to remember, therefore, that the dither frequency must be sufficiently higher than the missile control frequencies (error signal frequencies) so that the gain adaptive feature can be maintained. This imposes a severe lower bound on an acceptable dither frequency.

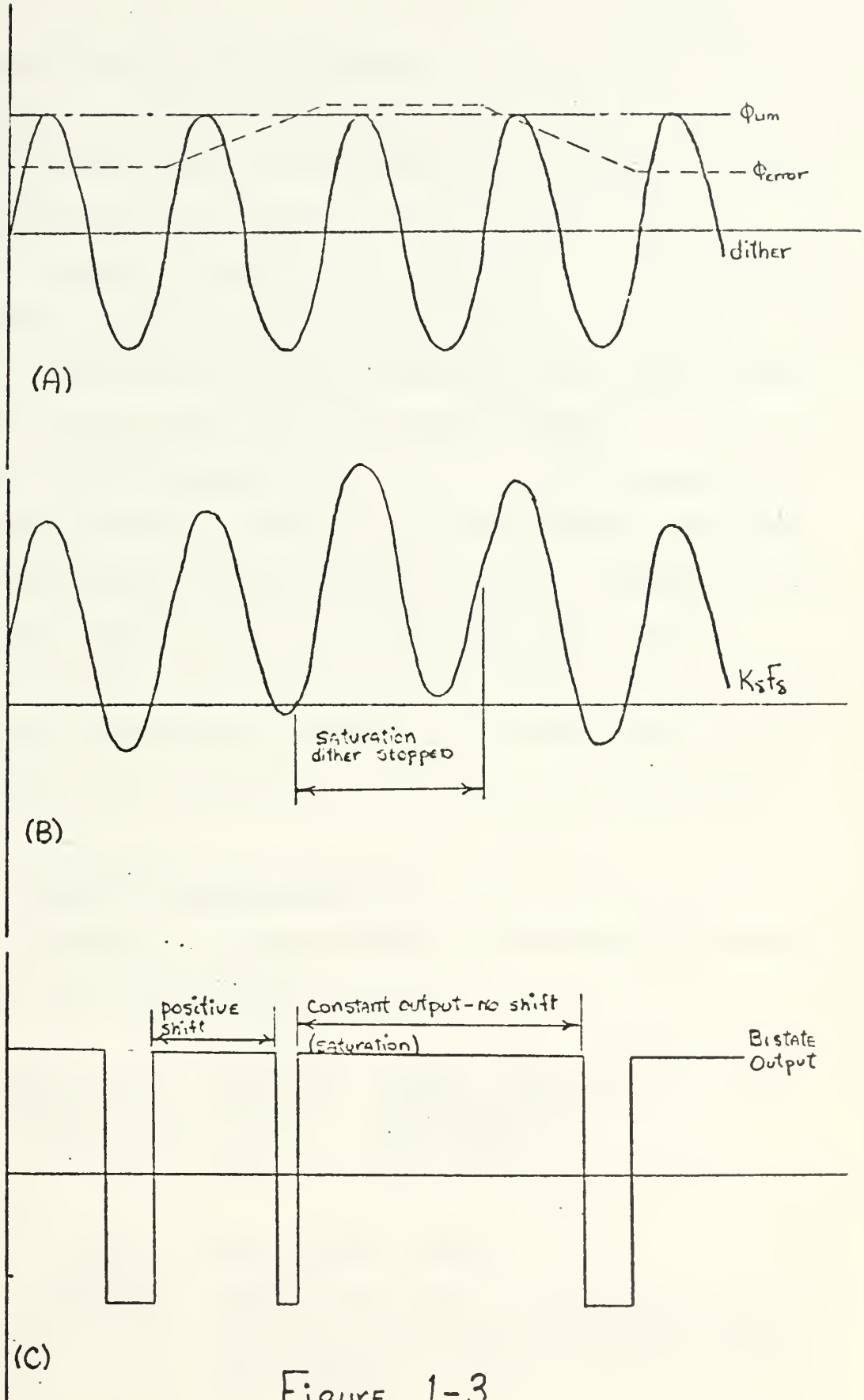

Φum
Φerror
dither
(A)
KsFs
Saturation
dither stopped
(B)
positive
shift
Constant output-no shift
(saturation)
Bistate
Output
(C)

Figure 1-3

E. DESCRIPTION OF LOOP COMPONENTS

Figure 1-1 can be divided into three major portions: the tail control surface actuators, the air frame, and the autopilot electronics package with associated sensors.

The original design work on the missile started with a given set of actuators and the air frame's equations of motion. The autopilot control equations were then written to drive the air frame with the given hardware.

Three major parameters are input or sensed within the roll control system. They are ϕ_c, the commanded roll angle, $K_\delta F_\delta$, the bistable element forcing function (commonly called "delta dot called for") and $\dot{\phi}'$, the tail roll rate as measured by the roll rate gyro. $\dot{\phi}'$ is integrated by the roll free gyro to produce the sensed missile roll angle ϕ'. ϕ_c and ϕ' are then differenced to obtain ϕ_e, the roll error signal.

1. Bistable Forcing Function

Equation 2 is the equation of the basic roll forcing function into the bistable switcher.

$$K_\delta F_\delta = \frac{(\phi' - \phi_c)_{Lim}}{K\phi} + .331\left[K\dot{\phi} + \frac{.151S\left(1+\frac{S}{1000}\right)\dot{\phi}'}{\left(1+\frac{S}{1000}\right)}\right] \qquad (2)$$

where: $K_\delta F_\delta$ = roll control signal

$(\phi' - \phi_c)$ = the electronically limited error signal. If the error signal is allowed to get too large, the loop will stop dithering and diverge

$\dot{\phi}'$ = roll rate output of the roll rate gyro

$K_\phi, K_{\dot{\phi}}$ are system constants

Preliminary analysis showed that high frequency resonances caused by the missile's elastic body bending mode were possible. To suppress signals at resonant frequencies and at the same time maintain the dither frequency, a cascaded set of high frequency ($\omega_{filter} > \omega_{dither}$) notch filters were added. The final form of the roll forcing function (Equation 2) becomes:

$$K_\delta F_\delta = G_F\left[\left(\frac{\dot{\phi}'-\dot{\phi}_c}{K_\phi}\right)_{Lim} + .331\left(K_{\dot{\phi}} + \frac{.151S\left(1+\frac{S}{1000}\right)}{\left(1+\frac{S}{1000}\right)} G_E G_{\dot{\phi}}\right)\dot{\phi}'\right] \qquad (3)$$

where

$$G_F = \frac{-K_\delta\left(1+\frac{S}{10}\right)}{S\left[1+\frac{2(.3)}{490}S+\left(\frac{S}{490}\right)^2\right]} \qquad (3a)$$

$$G_E = \frac{\left[1+\frac{2(.05)}{495}S+\left(\frac{S}{495}\right)^2\right]}{\left[1+\frac{2(.3)}{460}S+\left(\frac{S}{460}\right)^2\right]} \qquad (3b)$$

$$G_{\dot{\phi}} = \frac{\left[1+\frac{2(.05)}{755}S+\left(\frac{S}{755}\right)^2\right]\left[1+\frac{2(.05)}{755}S+\left(\frac{S}{755}\right)^2\right]}{\left[1+\frac{2(.1)}{680}S+\left(\frac{S}{680}\right)^2\right]\left[1+\frac{2(.4)}{680}S+\left(\frac{S}{680}\right)^2\right]} \cdot$$

$$\frac{\left[1+\frac{2(.05)}{1000}S+\left(\frac{S}{1000}\right)^2\right]}{\left[1+\frac{2(.7)}{1000}S+\left(\frac{S}{1000}\right)^2\right]} \qquad (3c)$$

2. Roll Rate Gyro

The transfer function for the roll rate gyro is

$$\frac{\dot{\phi}''}{\dot{\phi}'} = \left[\frac{1}{1 + \left(\frac{2\xi}{\omega}\right)s + \left(\frac{s}{\omega}\right)^2} \right] \tag{4}$$

where: $\dot{\phi}''$ = the gyro output signal

$\dot{\phi}'$ = missile roll rate

ξ = mechanical damping coefficient

3. Roll Free Gyro

The roll free gyro has a transfer function of unity and is considered linear over all normal operating ranges.

4. Tail Servo Transfer Function

The tail servo transfer function (using a rigid body model vice an elastic body model) can be described as

$$\frac{\gamma}{\gamma_c} = e^{-\frac{(\tau + \Delta\tau)s}{s}} \tag{5}$$

where: τ = the actuator time delay

$\Delta\tau$ = an approximation to the extra phase lag introduced by high frequency noise filtering, switch hystersis and aerodynamic plane lag due to the tail dithering.

5. Tail Surface Hinge Moment

Shown as a positive feedback path around the tail servo transfer function is $K_\ell K_T$. K_ℓ represents the hinge moment due to tail position. Aerodynamic loading forces acting on the tail control surfaces produce this hinge movement torque at the output of the tail actuators. The loop

is shown as a positive feedback loop, since the greater the effective tail angle, the greater the hinge moment torque produced; which, in turn, tends to increase the effective tail angle. K_T is a conversion factor, and is given as a system constant.

In this thesis, the effect of hinge moment relative to other system parameters was negligible; and in all simulations it was set to zero. Hinge moment was included in the model for completeness.

6. F_a and $F_{squeeze}$

F_a is the pitch control signal. It is produced in the pitch control loop, then summed with the roll control signal as an input into the bistable element. $F_{squeeze}$ is the control signal generated by the autopilot to compensate for the misalignment of the tail surfaces. Misalignment, resulting from mechanical tolerances and system slop in each of the four tail surfaces causes torques on the missile frame; counteracting torques may not produce roll or pitch, but will produce frame stress or squeeze.

7. Biases

The effect of biases in the roll loop is to cause the missile to assume a roll attitude other than that established at time of launch. Maintenance of roll orientation is essential to the guidance portion of the missile. The various components of the roll loops in addition to the pitch loop through F_a and the squeeze signal, $F_{squeeze}$, all introduce biases.

These loop biases are counteracted by the filter block:

$$F(s) = \left(\frac{1 + \frac{s}{10}}{s}\right) \tag{6}$$

The equivalent roll error of each contributor can be determined by solving for ϕ in the following simplified static roll control equation (7), with each bias substituted in terms of deg or deg/sec in the appropriate term.

$$s\dot{\delta} = K_L \left[\left(\frac{1 + \frac{s}{10}}{s}\right)\left(\frac{\phi'}{K\phi} + .331 K\phi \dot{\phi}' + K_\delta F_\delta\right)\right] + \dot{\delta}_{bias} \tag{7}$$

II. MODELING THE MISSILE

A. INTRODUCTION

There were two major areas of study in the research. First was the construction and testing of an accurate computer model to simulate the missile. Second was an investigation of an abnormal phenomenon experienced in the missile roll control loop called "roll wander." An explanation of this problem and a proposed solution will be presented in subsequent chapters.

B. SIMULATION

Due to the complexities of the missile control system and the inherent nonlinear operation introduced by the bistable element, it was decided to model the control system using a completely digital simulation. The model was therefore constructed using IBM's Digital Simulation Language (DSL). DSL was chosen for the model because under its format complex transfer functions could be described by single blocks with accompanying data statements. This eliminated the need for writing state variable equations for as high as eighth order systems, and thereby significantly reduced the amount of coding required to model the missile. A listing of the DSL program with a more detailed commentary on the operation of the DSL program is provided in Appendices A and B.

Shown in Figure 2-1 is the missile roll control loop as it was arranged for encoding into DSL. The letter names for the different function blocks correspond to the variable names used in the simulation program. The DSL function TRNFR used to represent the various transfer functions imposes one restriction. It requires that the polynomial in the denominator be of a greater order than the polynomial in the numerator. The reason for this is because the TRNFR block determines the correct output response by solving the corresponding system of first order differential equations. This requires the polynomial in the denominator to have the highest order to allow for the construction of the correct first order system. To accommodate this restriction, certain blocks (B and HP) were formed by combining other blocks (as seen by a comparison of Figure 1-1 and Figure 2-1). This altered the appearance of the block diagram, but not the behavior of the loop.

Block D represents the mechanical system's response time to the bistable element's output signal. A finite time occurs between the electrical command and the mechanical movement of the tail control surfaces. This was modeled in the computer program by the DSL function block DELAY. DELAY is a dead-time function. Sample points of the input expression are saved at specified intervals. The output is set equal to zero until the delay interval expires at which time the output has the value of the input at the start of the delay. If necessary, linear interpolation is used between sample points.

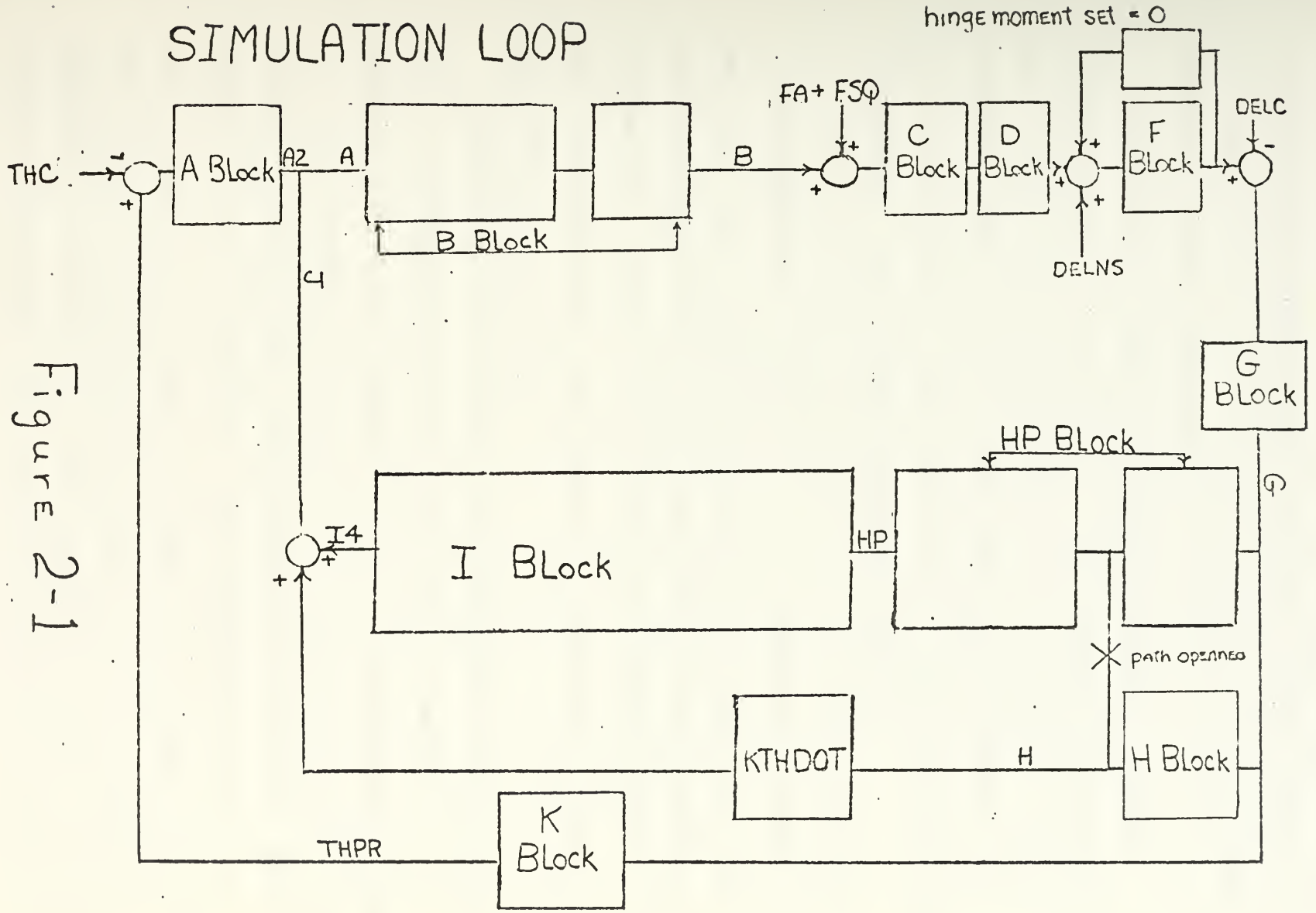

SIMULATION LOOP
hinge moment set = 0
THC
A Block
A2
A
B
B Block
FA+ FSQ
C Block
D Block
F Block
DELC
DELNS
G Block
HP Block
HP
I4
I Block
path openned
KTHDOT
H
H Block
K Block
THPR

Figure 2-1

This ability, gained using an all digital simulation to accurately and easily adjust the delay interval, was one of the major factors in choosing the digital simulation over an equivalent analog simulation.

The bistable element was modeled using a standard FORTRAN logical IF statement. The bistable element has no zero point, only full positive and full negative. The logical IF was written so that when $K_\delta F_\delta$ was greater than zero, the bistable output was positive. When $K_\delta F_\delta$ was zero or negative, the bistable output was negative.

The aerodynamic gain constants G and F were calculated at the beginning of each run. The parameters necessary to calculate these values and the parameters specifying the other systems constants were specified at the beginning of a given run. These constants could be changed during or at the end of the run, thus giving the capability of multiple runs within one DSL job.

Difficulty was initially encountered due to the step size used. The TRNFR blocks are comprised of a series of integrators simulated by DSL by a call to an integration routine of a type specified by the user. To conserve computer time, a fixed interval fourth order Runga Kutta method was used. However, it is a property of this integration routine as well as many other numerical integration routines, that if the frequency components of the function to be integrated approach the frequency of the integration routine, specified by the routine step size, the integration method produces a

divergent error and becomes unstable. Such was the case with this program. Initially, a step size of 0.005 was used. Problems were encountered with the inclusion of the high frequency notch filters. The notch filter represented by a denominator of:

$$1 + \frac{2.0(0.7)S}{1000} + \left(\frac{S}{1000}\right)^2 \qquad (8)$$

has a center frequency of 159 hz.

Similarly, the notch filter represented by:

$$1 + \frac{2.0(0.9)S}{680} + \left(\frac{S}{680}\right)^2 \qquad (9)$$

has a center frequency of 108 hz. Using a step size with an integration frequency of only 200 hz was insufficient, and caused divergence.

Optimal step size was obtained by testing the loop using a variable step size Runga Kutta method with different error limits specified. By specifying a tolerance of 0.001 absolute error an average integration step size of 0.0004 was obtained. Thus, a frequency of 2500 hz was sufficiently high to avoid numerical integration problems. During the actual runs, a final step size of 0.0001 was used.

C. TESTING THE MODEL

Except for the problem of determining the program integration step size, the only difficulties encountered with

the initial simulation runs were those of writing the standard program logic and debugging routine syntactical errors. Once constructed, the model was tested under normal operating conditions to ensure that it behaved the same as the real missile. The model was allowed to "fly" with no input (Figure 2-2). The initial jump noted is the transient behavior of the loop. Since the parameters of the loop were continually being changed from one run to the next, initial conditions of the integrators were not calculated; rather all integrators were set initially to zero, and the transient response accepted. The low amplitude oscillation is the bistable dithering action. The dither appeared in the position loop because of the step size of integration used. By decreasing step size, the amplitude of the dither could be reduced. However, decreasing step size proportionately increased run time. A compromise was reached using a step size of 0.0001.

The closed loop transfer function for a roll command (ϕ/ϕ_c) obtained by writing the loop equation for Figure 1-1, (with K = 0) is:

$$\frac{\phi}{\phi_c} = \frac{K_\delta K_\iota e^{-TS}\left(\frac{\dot{\phi}}{S}\right)\left(1+\frac{S}{10}\right)}{K_\phi S^3\left[1+\frac{2(.3)}{470}S+\left(\frac{S}{470}\right)^2\right]+K_\phi K_\delta K_\iota e^{-TS}\left(\frac{\dot{\phi}}{S}\right)\left(1+\frac{S}{10}\right)} \quad (10)$$

$$\frac{}{.331\,G_R S\left[K_{\dot{\phi}}+\frac{.151S\left(1+\frac{S}{100}\right)}{1+\frac{S}{2000}}G_\epsilon G_{\dot{\phi}}+K_\delta K_\iota e^{-TS}\left(\frac{\dot{\phi}}{S}\right)\left(1+\frac{S}{10}\right)\right.}$$

where: G_R = transfer function of the roll rate gyro

$G_E G_{\dot{\phi}}$ = torsional notch filter

Applying the final value theorem for a step input ($\phi_c = \phi_c/s$), to Equation (10) gives:

$$\lim_{t\to\infty} \phi = \phi_c \frac{K_\xi K_L e^{-TS}\left(\frac{\dot{\phi}}{\xi}\right)\left(1+\frac{S}{10}\right)}{K_\delta K_L e^{-TS}\left(\frac{\dot{\phi}}{\xi}\right)\left(1+\frac{S}{10}\right)} \qquad (11)$$

Similarly, the closed loop expression for a tail induced torque (ϕ_c/δ_c), also obtained from Figure 1-1 is:

$$\frac{\ddot{\phi}}{\delta_c} = S\frac{\dot{\phi}}{\delta_c}$$

$$= \frac{-K_\phi S^4\left(\frac{\dot{\phi}}{\xi}\right)\left[1+\frac{2(.3)}{470}S+\left(\frac{S}{470}\right)^2\right]}{K_\phi S^3\left(1+\frac{2(.3)}{470}S+\left(\frac{S}{470}\right)^2\right)+K_\phi K_\delta K_L e^{-TS}\left(\frac{\dot{\phi}}{S}\right)\left(1+\frac{S}{10}\right)}$$

$$\cdot\left(.331\, G_R S\left(K\dot{\phi}+\frac{.151\left(1+\frac{S}{100}\right)}{1+\frac{S}{2000}}G_{\dot{\phi}}\right)+K_\delta K_L e^{-TS}\left(\frac{\phi}{\delta}\right)\left(1+\frac{S}{10}\right)\right) \qquad (12)$$

Then for a step input (δ_c = const/s), and applying the final value theorem to Equation (12)

$$\lim_{t \to \infty} \ddot{\phi} = 0 \tag{13}$$

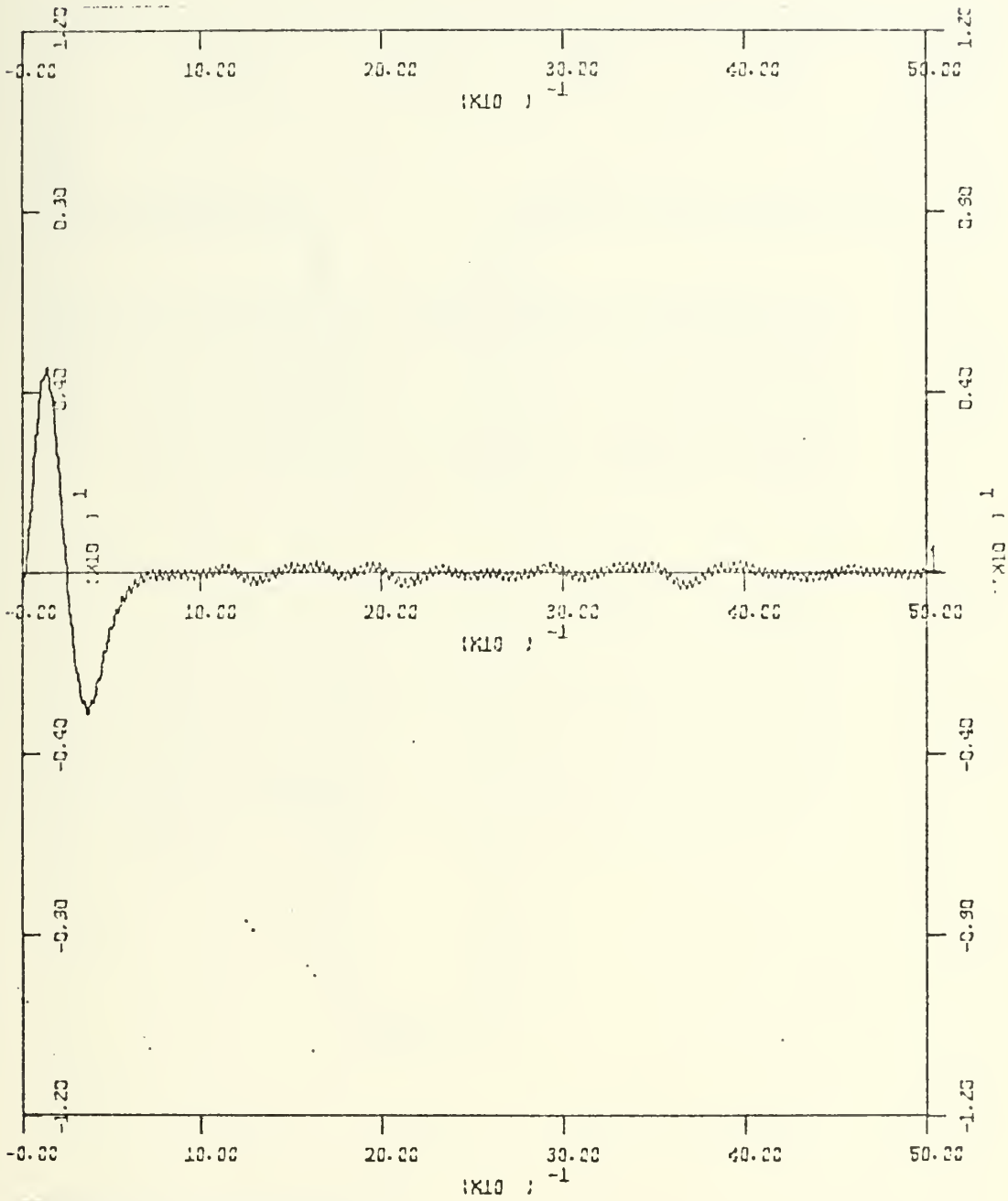
The model was thus tested using a step input of -45° for ϕ_c and a step input of 3° for δ_c. The response to the step input of ϕ_c is shown in Figure (2-3). For a roll command or a roll error signal, the missile will roll and maintain the desired angle. The response of the model compared favorably with that of an actual missile. The measured time constant of the missile was approximately 0.21 seconds over the various flight conditions. The measured response of the model was 0.19 seconds over the same flight conditions.

The transient response time to a step torque, equivalent to a step input (δ_c) of 3° was approximately 0.5 seconds for the actual missile and 0.62 seconds for the model (shown in Figure 2-5).

Tests were conducted on the missile model under two sets of flight conditions covering the limits of normal operation. Flight condition No. 15 represented normal altitude and velocity, whereas flight condition No. 24 represented low altitude and normal velocity. The self-adaptive nature of the control loop can be seen in Figures 2-4 through 2-10. The amplitude of $K_\delta F_\delta$ changes according to the aerodynamic conditions, but the response (ϕ/ϕ_c or ϕ/δ_c) remains constant.

FIGURE 2-2

ϕ' vs TIME with No Inputs

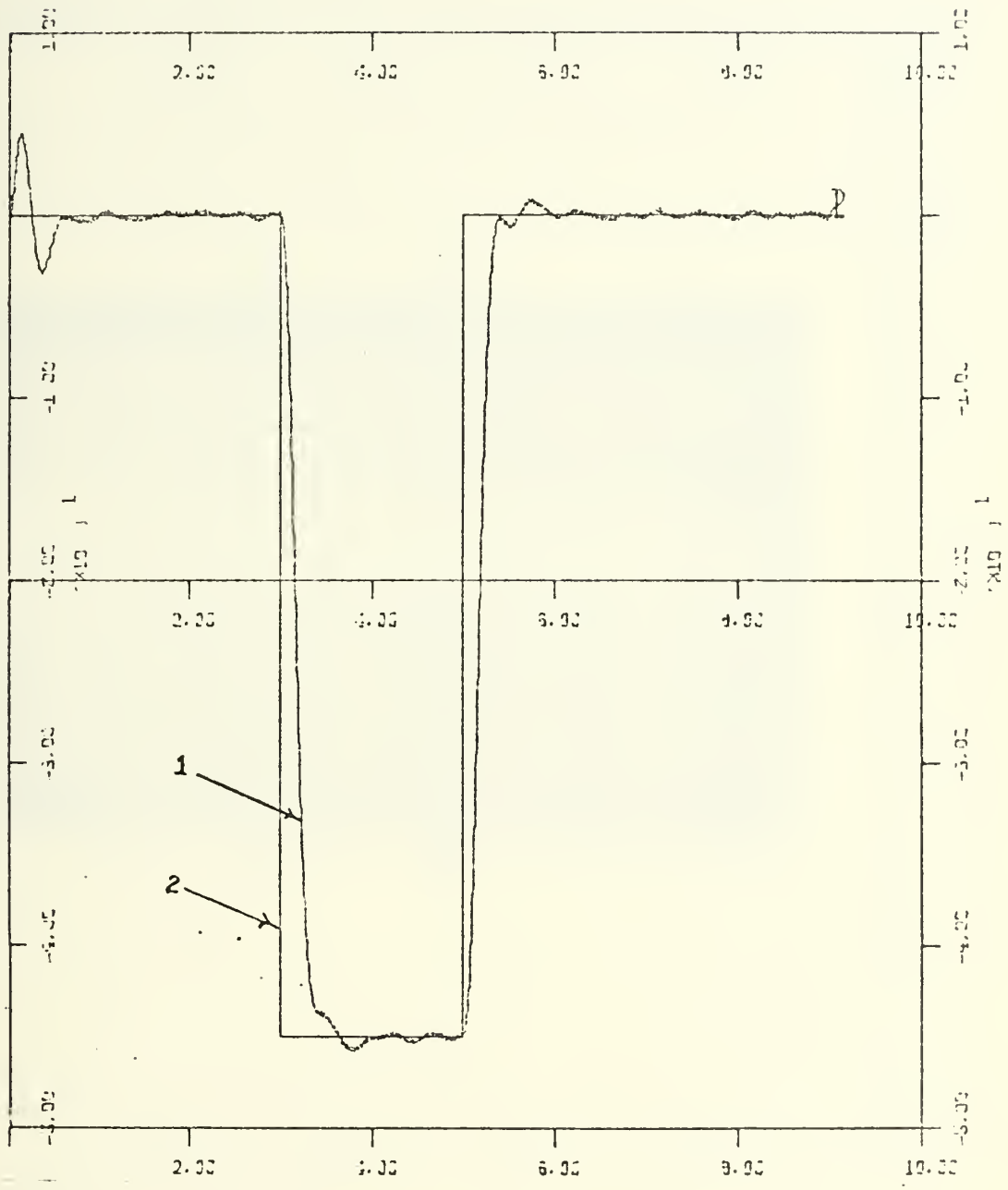


Xscale: 1.0 sec/inch
Yscale: 4°/inch

Note: Initial excursion is due to the initial conditions used in the simulation.

FIGURE 2-3

ϕ' vs TIME with -45° step input @ t=3.0 sec
(flight condition #15)

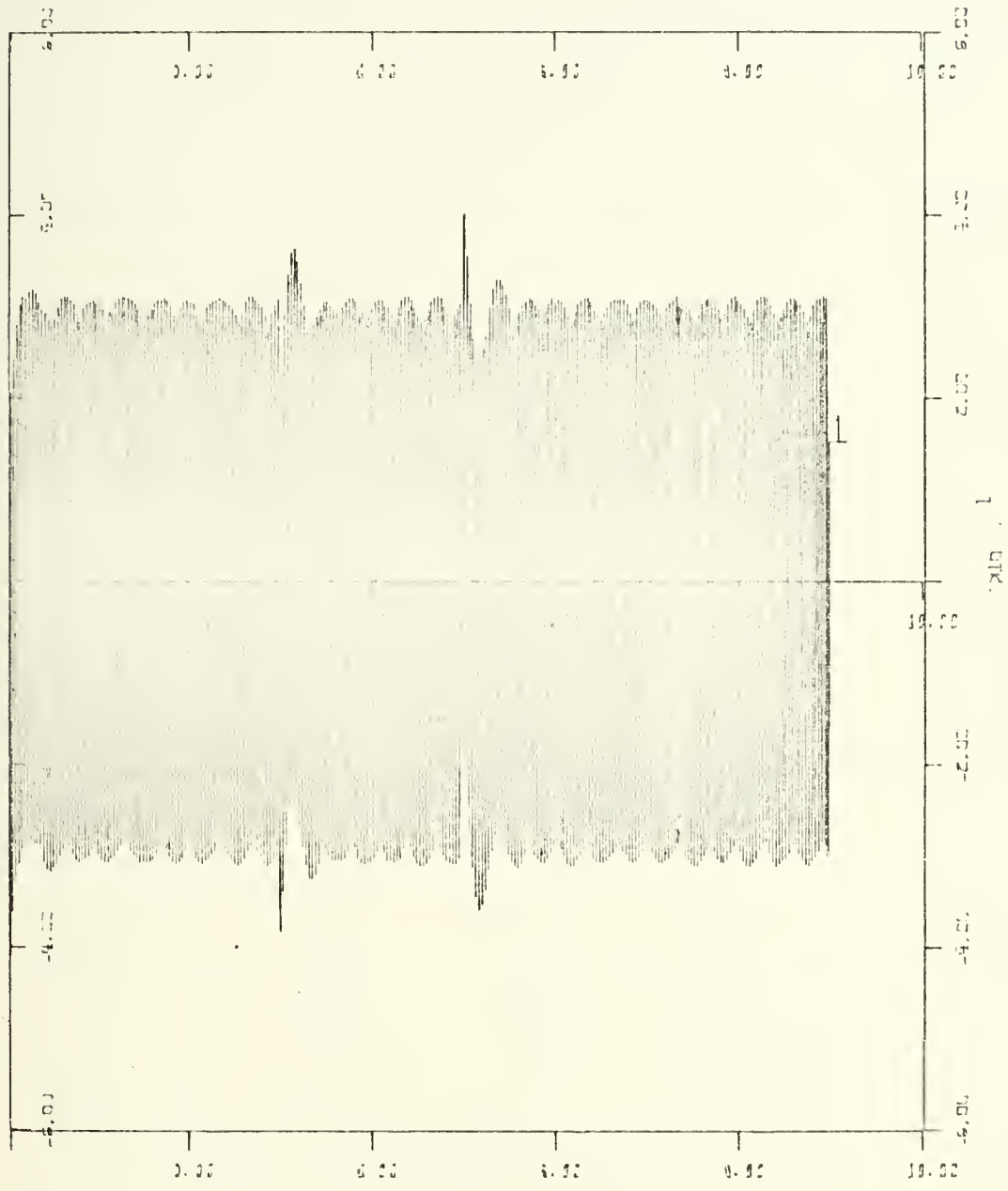


Xscale: 2.0 sec/inch
Yscale: 10°/inch

Curve #1 ϕ'
Curve #2 Step Input

FIGURE 2-4

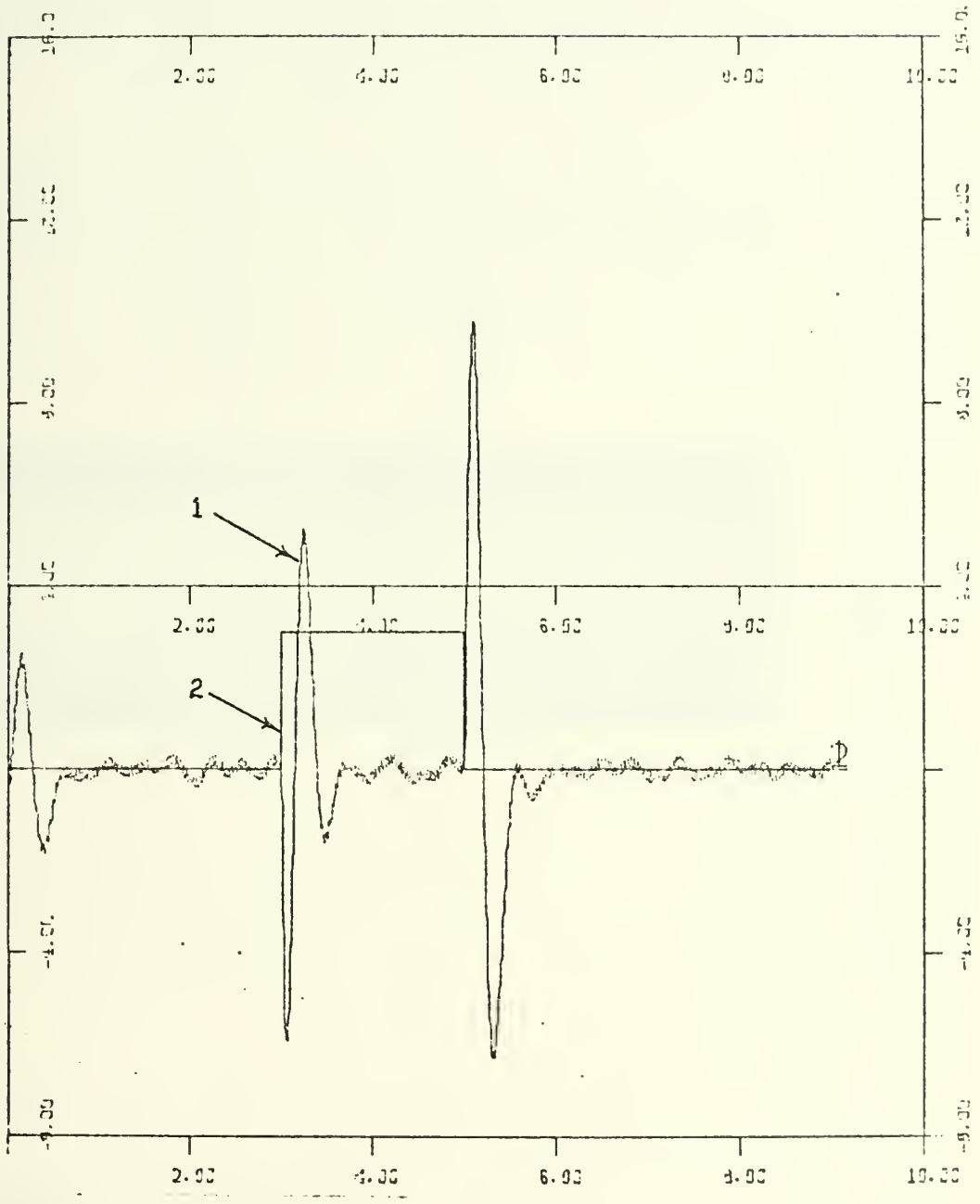
K_{δ_F} vs TIME with -45° step input @ t=3.0 sec.
($\delta_{flight\ condition\ \#15}$)



Xscale: 2.0 sec/inch
Yscale: 20°/sec/inch

FIGURE 2-5

ϕ' vs TIME with +3° step torque (δ_c) at t=3.0 sec.
(flight condition #15)

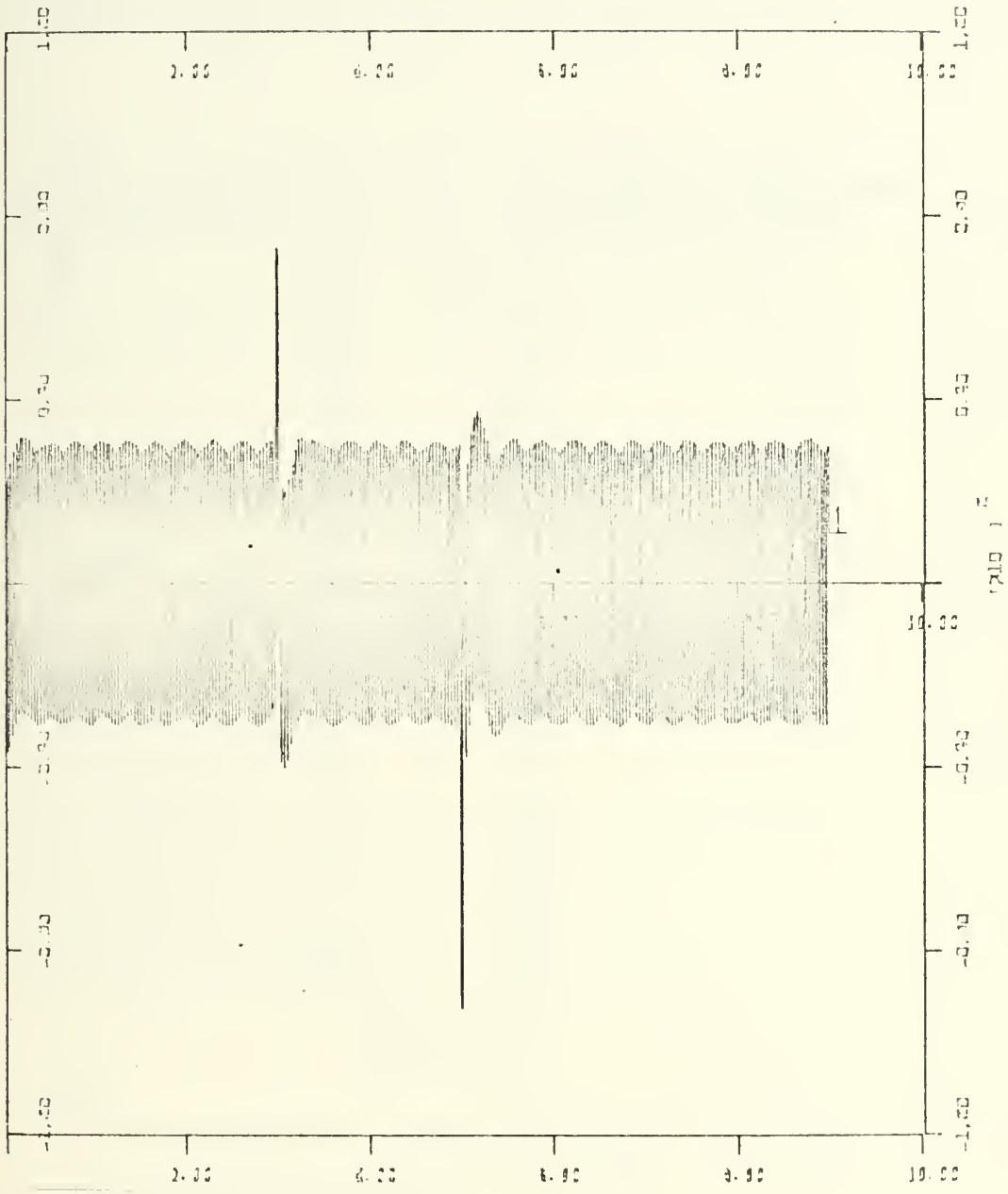


Xscale: 2.0 sec/inch
Yscale: 4°/inch

Curve #1 ϕ'
Curve #2 δ_c

FIGURE 2-6

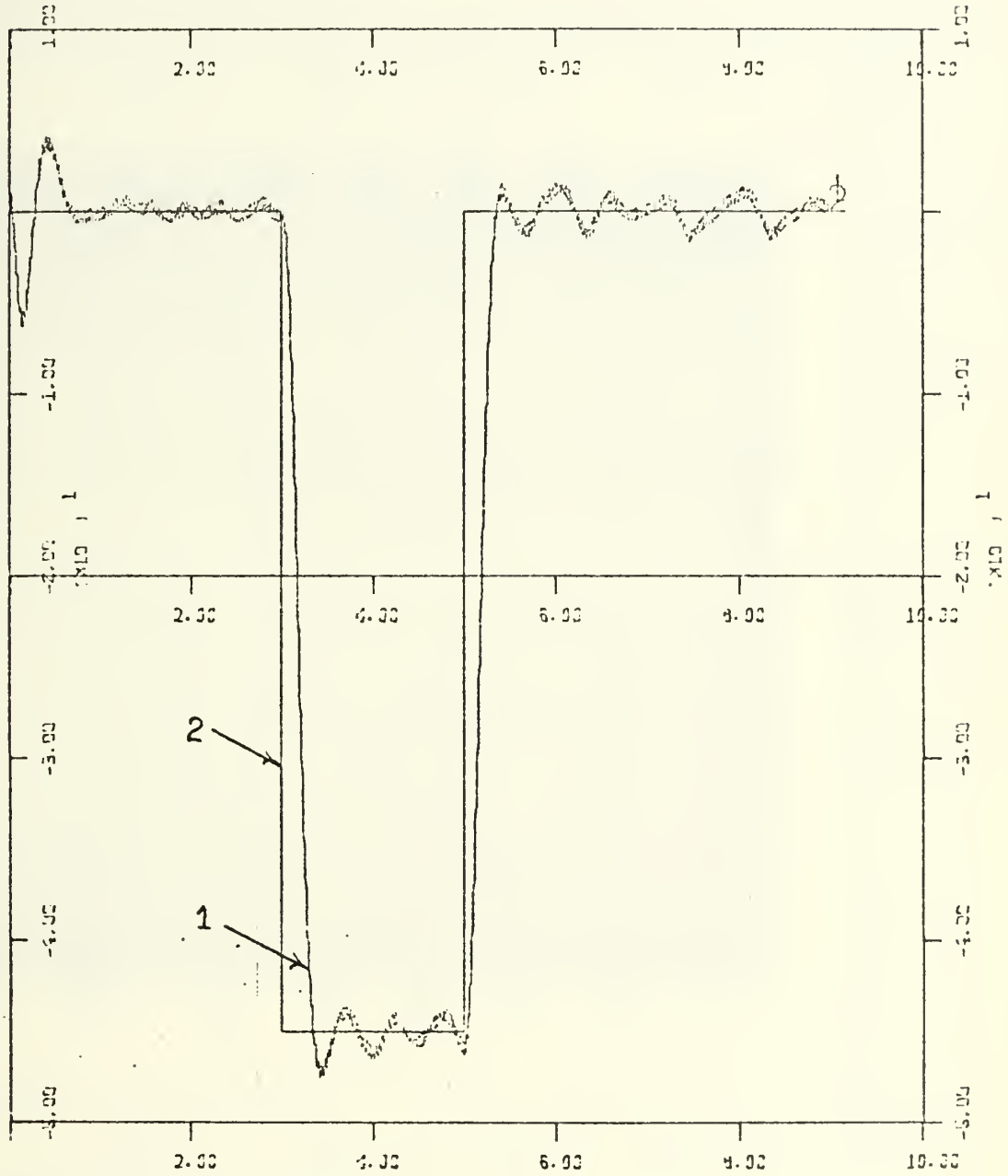
$K_{\delta}F_{\delta}$ vs TIME with +3° step torque (δ_c) at t=3.0 sec.
($\delta_{flight\ condition\ \#15}$)



Xscale: 2.0 sec/inch
Yscale: 40°/sec/inch

FIGURE 2-7

ϕ' vs TIME with a -45° step input at t=3.0 sec.
(flight condition #24)

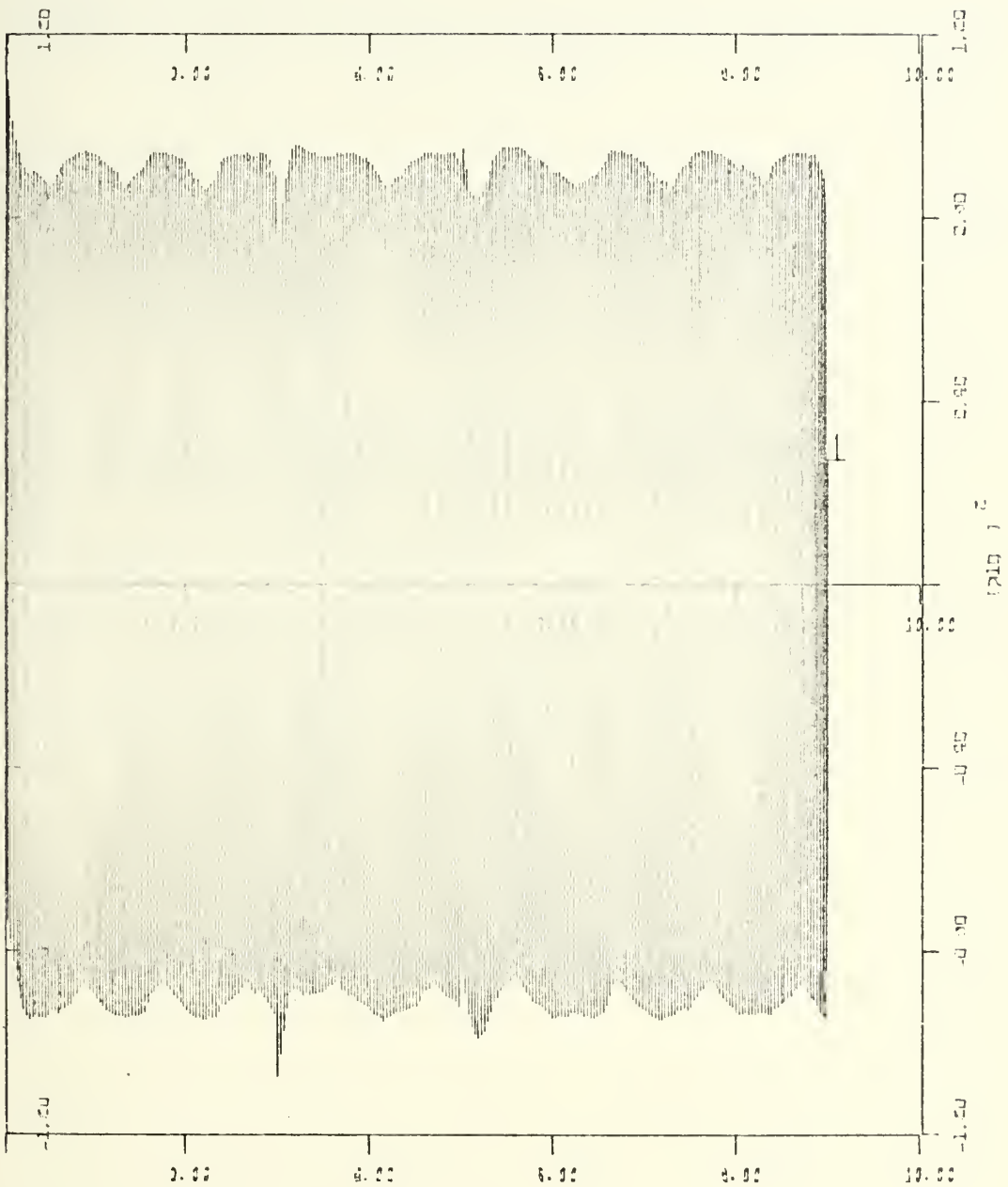


Xscale: 2.0 sec/inch
Yscale: 10°/inch

Curve #1 ϕ'
Curve #2 Step Input

FIGURE 2-8

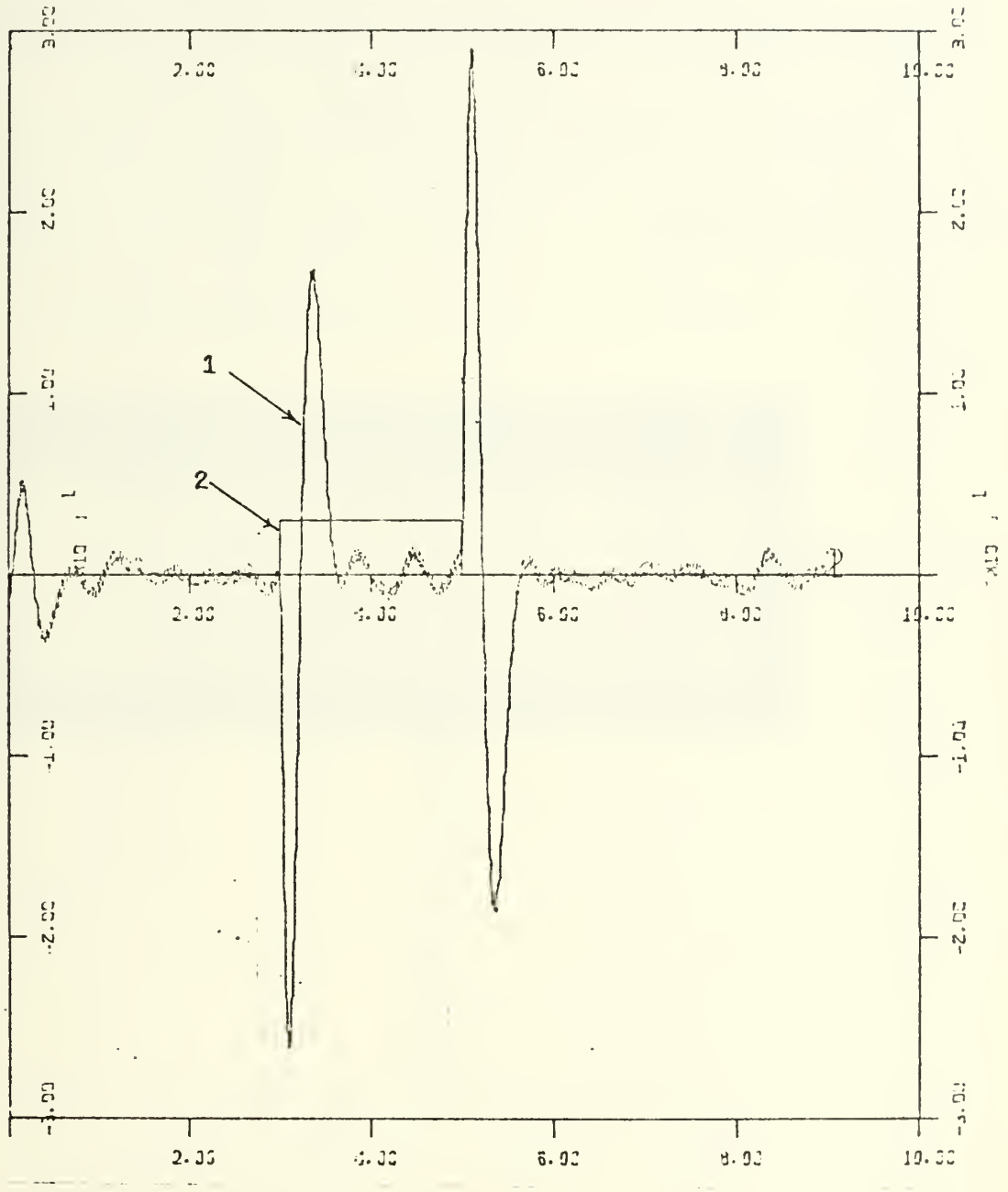
$K_\delta F_\delta$ vs TIME with a -45° step input at t=3.0 sec.
($^\delta$flight condition #24)



Xscale: 2.0 sec/inch
Yscale: 40°/sec/inch

FIGURE 2-9

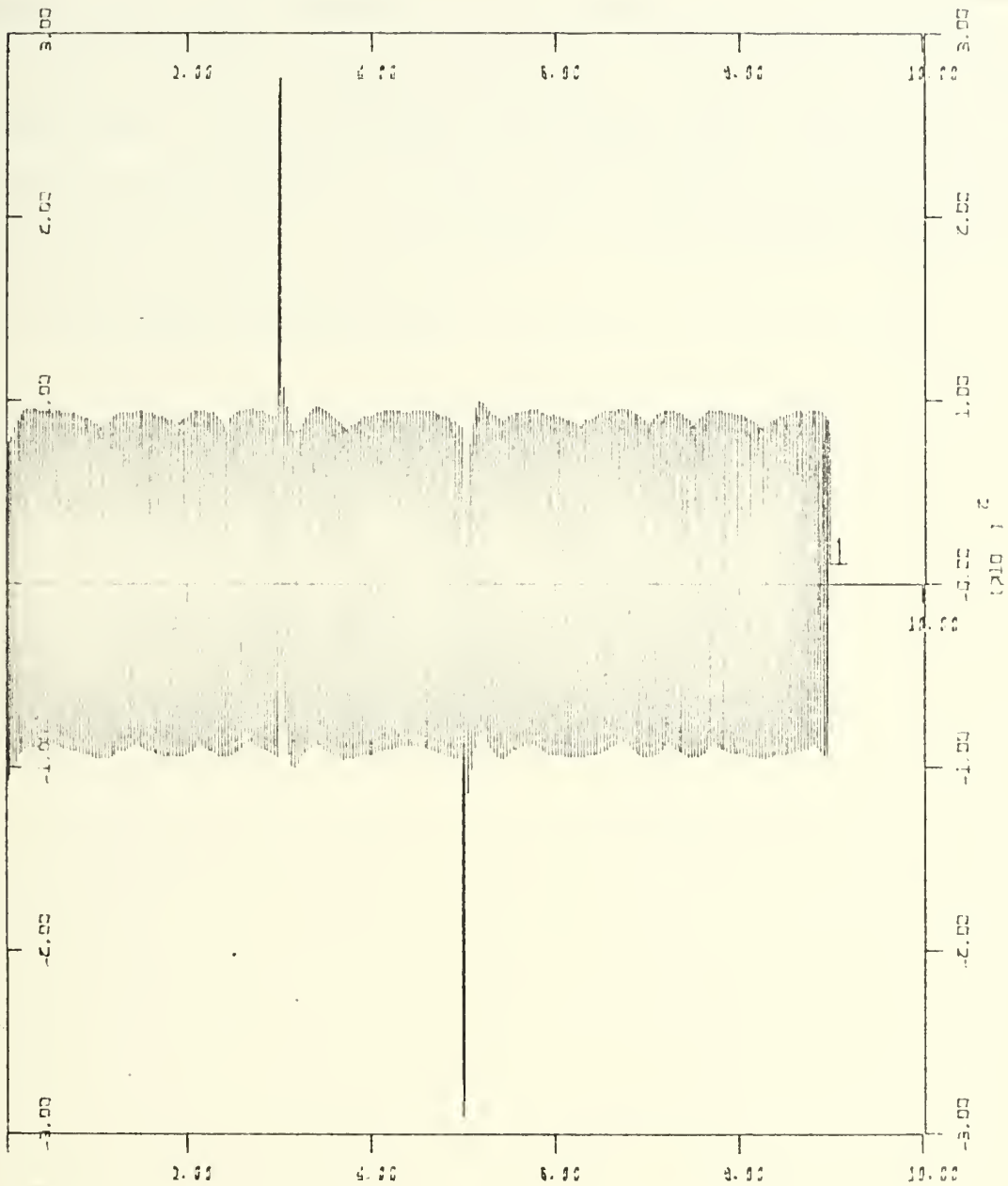
ϕ' vs TIME with a +3° step torque (δ_c) at t=3.0 sec.
(flight condition #24)



Xscale: 2.0 sec/inch
Yscale: 10°/inch

FIGURE 2-10

$K_{\delta} F_{\delta}$ vs TIME with a +3° step torque (δ_c) at t=3.0 sec.
(δ_{flight} condition #24)



Xscale: 2.0 sec/inch
Yscale: 100°/sec/inch

D. CONCLUSIONS

The simulation flights shown in Figure 2-2 through Figure 2-10 were compared to the results of test flights performed by the civilian contractor. On the basis of these comparisons, the conclusion was reached that the model was an accurate representation of the actual missile control system and that the model could therefore be used in the study of the roll wander phenomenon.

III. ROLL WANDER

A. INTRODUCTION

As stated earlier, the two major areas of research were the construction and testing of an accurate computer simulation for the missile and the study of the phenomenon of "roll wander."

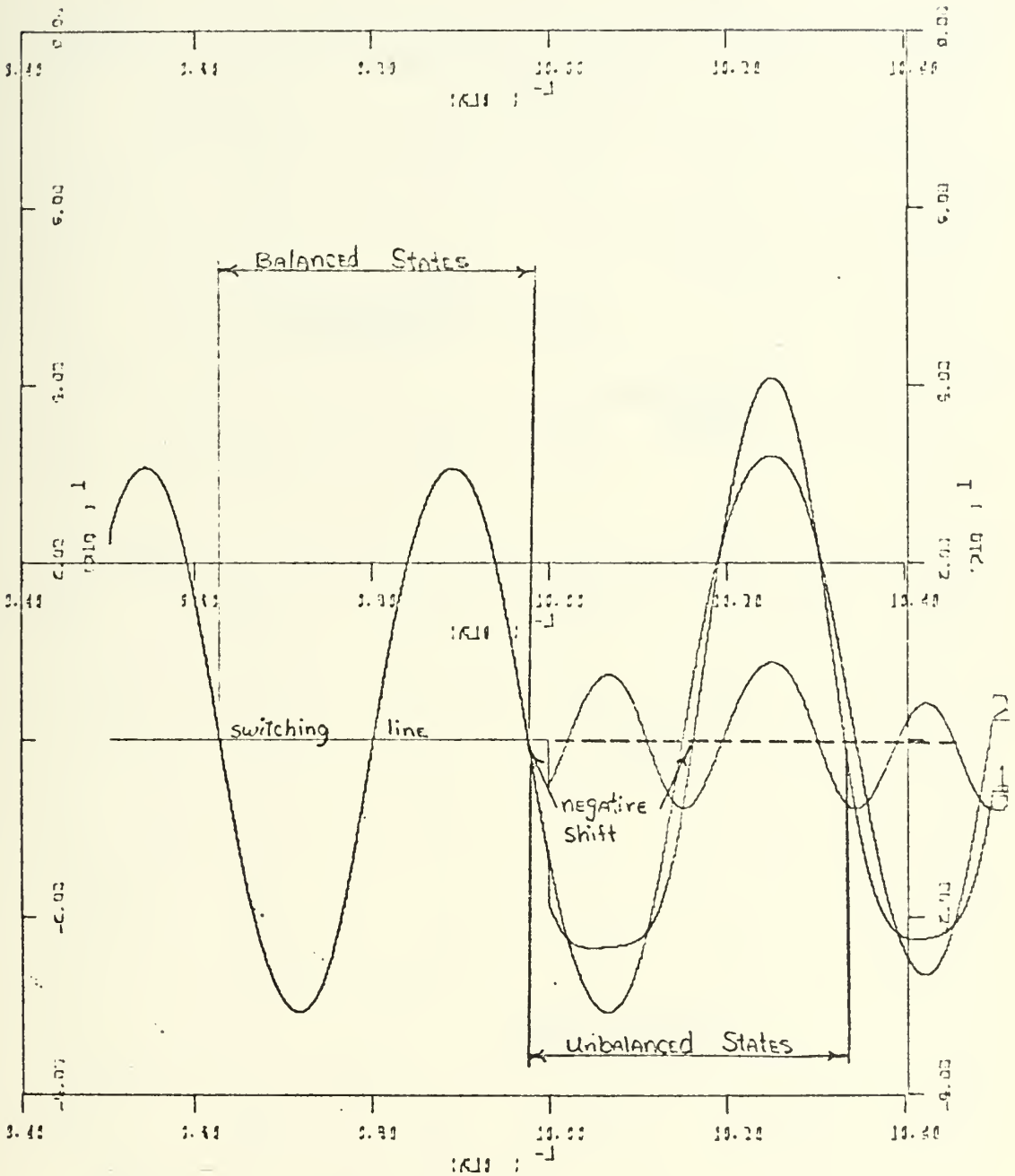
This undesired roll loop excitation or roll wander results from noise originating in the tail control surface actuators. This noise is sensed by the roll rate gyro and is thus introduced by the roll rate gyro into the rate feedback loop as a roll rate error signal. The production of roll wander then occurs in one of two fashions.

B. LOW AMPLITUDE ROLL WANDER

The noise signal introduced is enhanced by the effects of the roll loop nonlinearities. The effect of these nonlinearities is to generate harmonics of the dither frequency. These harmonics in turn add with the fundamental dither frequency to produce a distorted wave form of a frequency equal to the fundamental frequency but with zero crossings within one cycle unevenly spaced (Figure 3-1). Since the bistable element is triggered only by these zero crossings, the average frequency of the bistable remains the same; but the time in its two states becomes unbalanced (Figure 3-2). The neutral position of the tail surfaces is the integration of the bistable element output. Therefore, the tail control surfaces

FIGURE 3-1

$K_\delta F_\delta$ vs TIME Showing Harmonic Addition
($^\delta$Harmonic commences at t=1.0 sec.)

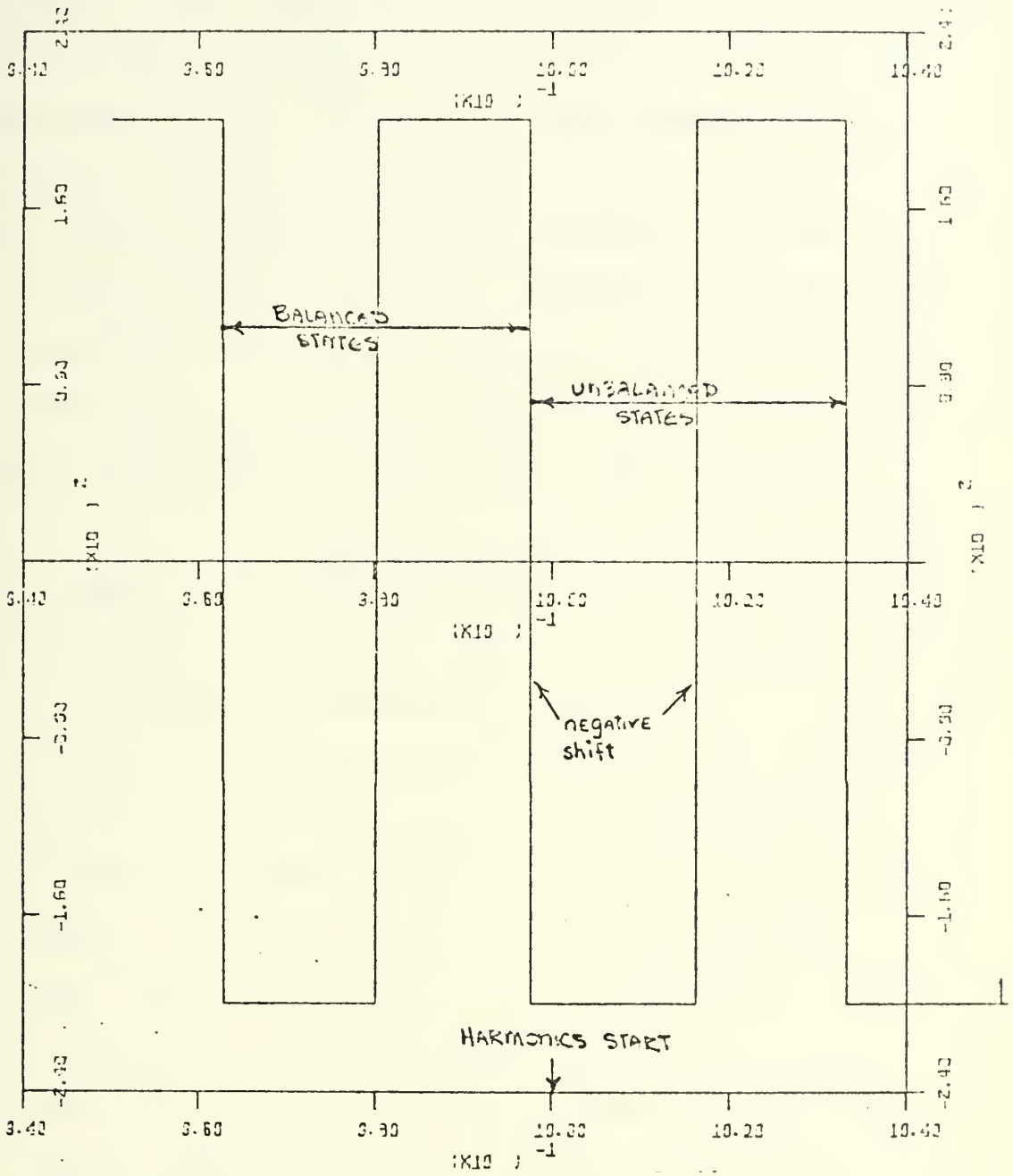


Xscale: 0.02 sec/inch
Yscale: 20°/sec/inch

Curve #1 Resultant distorted waveform
Curve #2 Fundamental dither frequency
Curve #3 Harmonic frequency

FIGURE 3-2

Bistable Output
(Harmonics Commence at t=1.0 sec.)

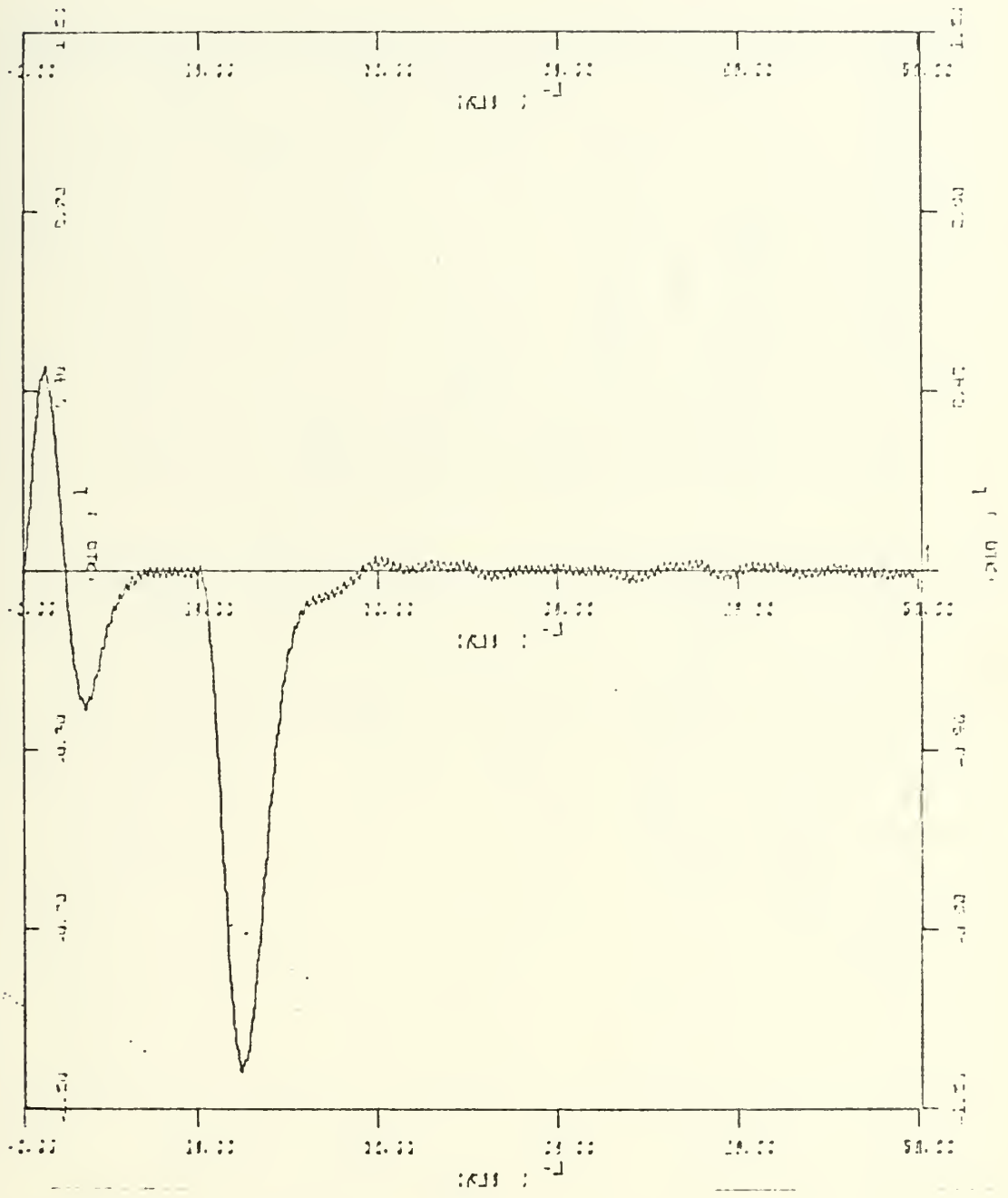


Xscale: 0.02 sec/inch
Yscale: 80°/sec/inch

begin to move off of their zero positions and the missile rolls. Since no roll command was introduced into the loop, a resultant dc error signal is generated in the roll position loop (outer loop). This dc signal causes the input to the bistable element to be shifted accordingly above or below the zero axis. Since the distortion of the waveform is caused by harmonic addition, and the fundamental frequency of the bistable remains unchanged, a shifted position exists for the waveform where the zero crossings once more become equal. The result is that the missile initially rolls off zero upon introduction of the harmonic, then rolls back (Figure 3-3). If the phase of the harmonic wave is shifted rapidly enough to repeatedly offset the resultant waveform, the missile will begin to wander in roll at a frequency of 1.2 to 2.5 hz trying to compensate (Figure 3-4). In the actual missile, uncorrelated white noise from the tail control actuators produces this phenomenon. The broad band noise contains components at the harmonic frequencies of the dither signal. Since this noise is random uncorrelated noise, there exists no coherence in the phase of the various spectral components. Therefore, the action of the noise is the same as the addition of the fundamental dither frequency with a series of harmonic frequencies shifting rapidly in phase. Noise with very low amplitude, only 10% of the amplitude of the signal (SN = +20db) is sufficient to cause the onset of roll wander (Figure 3-5).

FIGURE 3-3

ϕ' vs TIME Showing the Effect of a Fixed Phase Harmonic (Harmonic Commences at t=1.0 sec.)

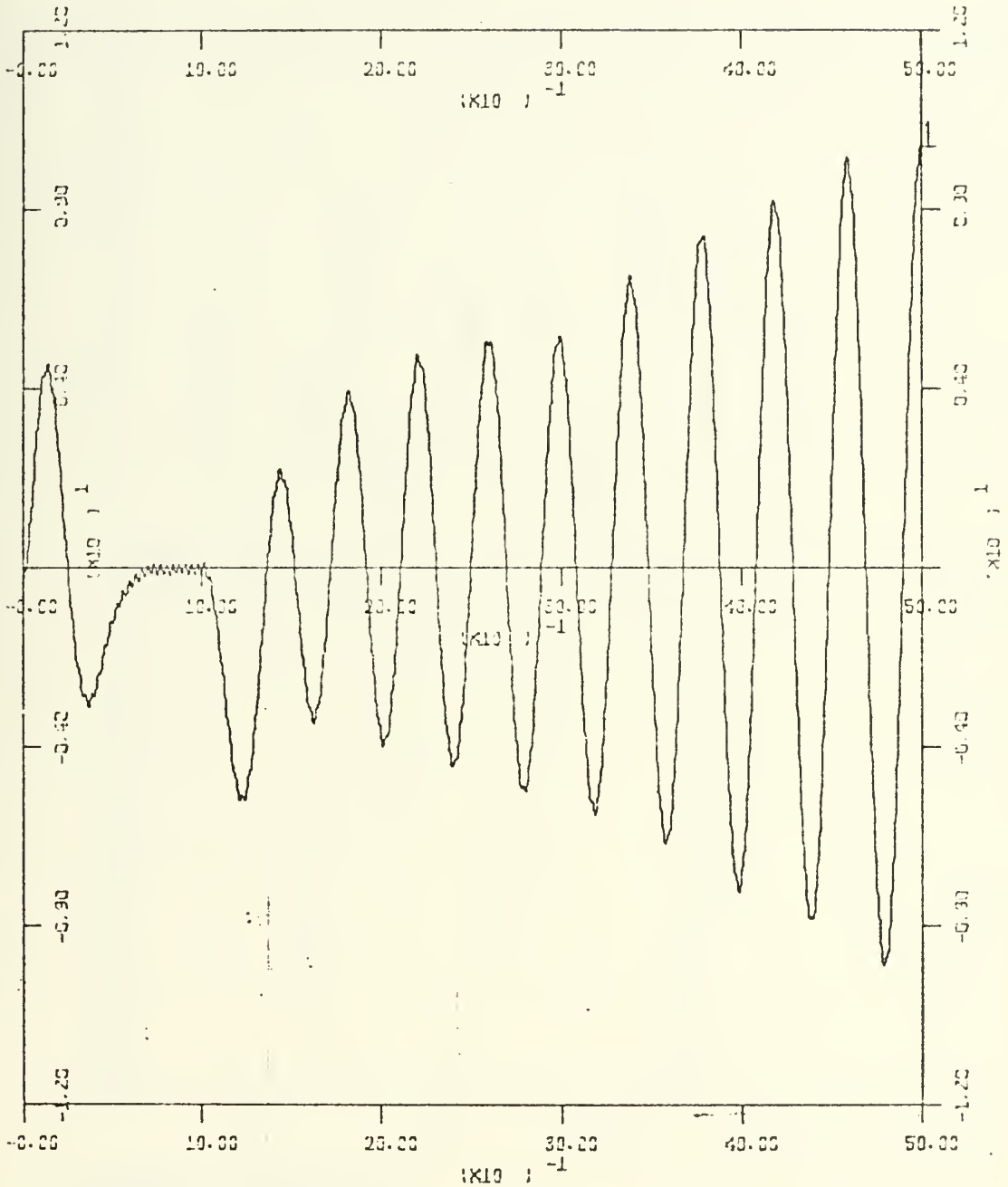


Xscale: 1.0 sec/inch
Yscale: 4°/inch

Note: Initial excursion is due to the initial conditions used in the simulation.

FIGURE 3-4

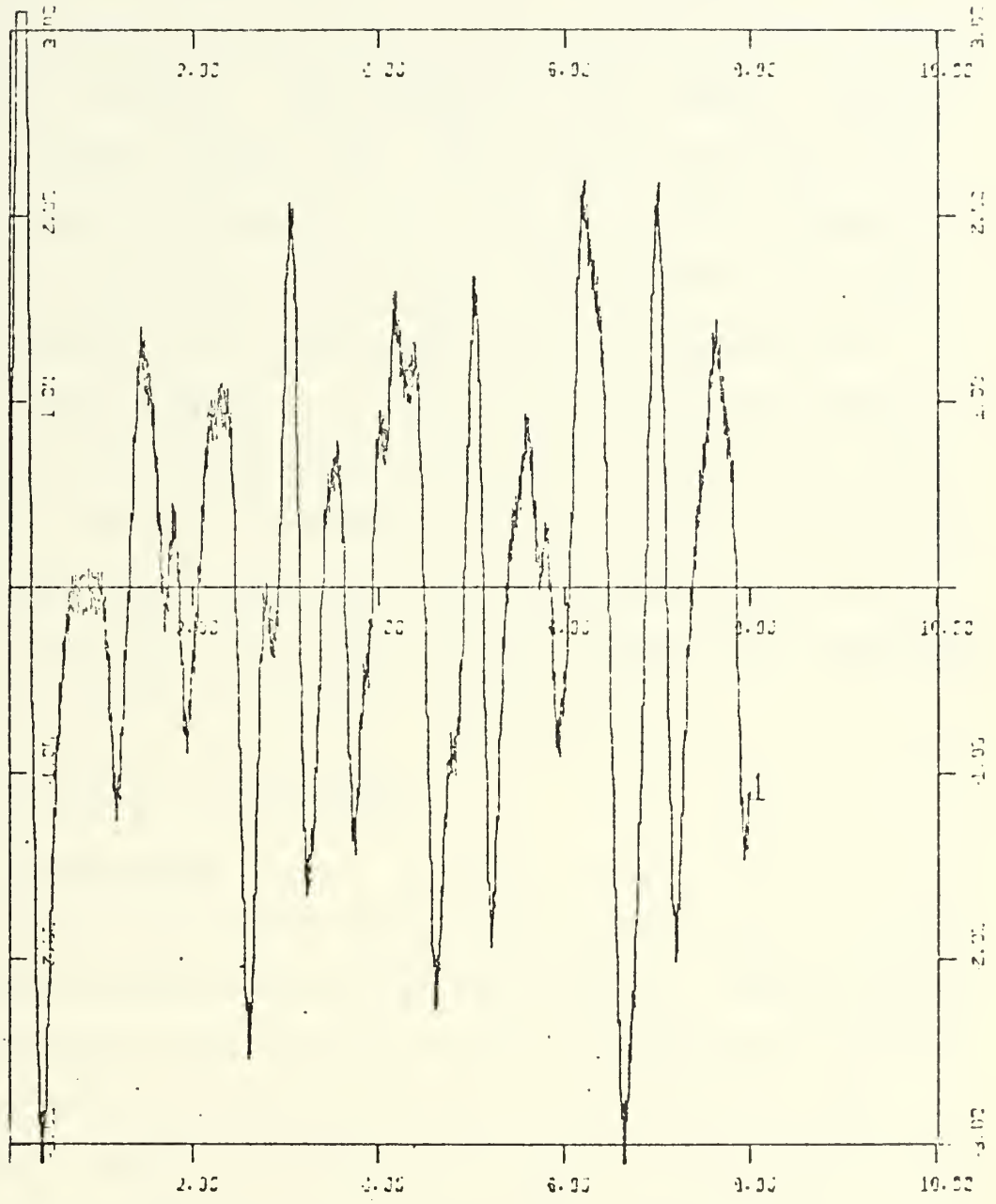
ϕ' vs TIME Showing the Effect of a Varying Phase Harmonic
(Harmonic Commences at t=1.0 sec.)



Xscale: 1.0 sec/inch
Yscale: 4°/inch

FIGURE 3-5

ϕ' vs TIME Showing the Effect of Adding White Noise (constant level) Noise Commences at t=0.7 sec.



Xscale: 2.0sec/inch
Yscale: 1°/inch

Note: Initial excursion is due to the initial conditions used in the simulation.

C. LARGE AMPLITUDE ROLL WANDER

As the noise amplitude increases, the amplitude of the wander increases, since the harmonic components of the noise signal also increase. The frequency of the wander, however, remains constant (Figure 3-6). The action of the noise is to reduce the low frequency roll loop gain. The decrease in the low frequency roll loop gain will decrease the roll system damping in the closed loop response, resulting in larger wander excursions. If the noise gets large enough, the bistable switch gain may be decreased to where the roll system will enter into a sustained low frequency oscillation of extremely large magnitude. The dithering action of the loop is stopped and the entire system operates in a saturated limit cycle mode of operation (Figure 3-7).

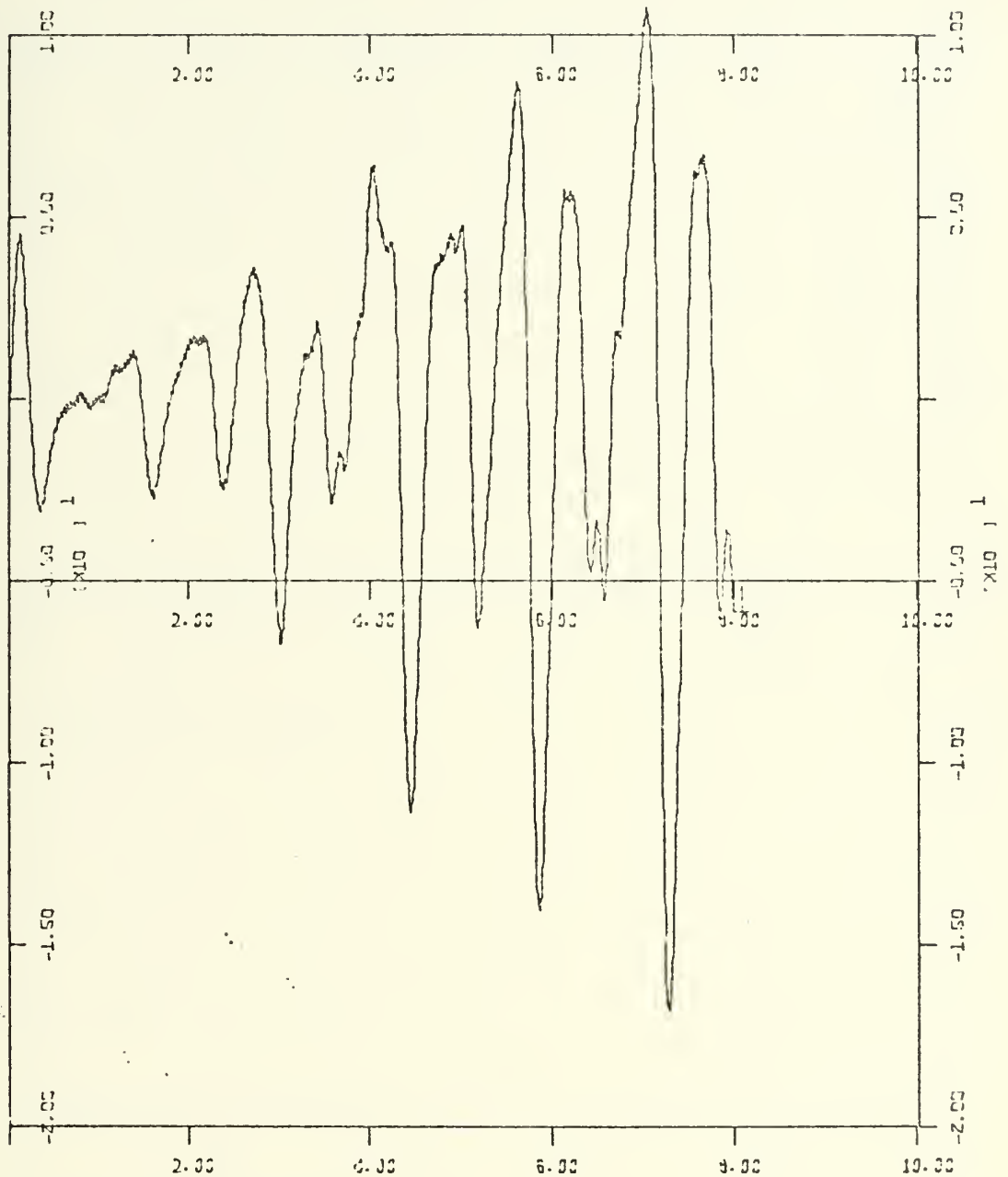
D. SIMULATING THE ROLL WANDER

1. Noise Source

One of the more difficult problems was that of accurately simulating the roll wander phenomenon. Analog simulations performed by the civilian contractor had been performed using 400 hz band limited gaussian noise. To produce this digitally, a FORTRAN subroutine was written utilizing a random number generator and a digitally simulated 400 hz low pass filter to generate the appropriate band limited noise. The digital simulation of the transfer function of a low pass filter requires the integration of all data points to produce the filtered output. Therefore, to preclude the

FIGURE 3-6

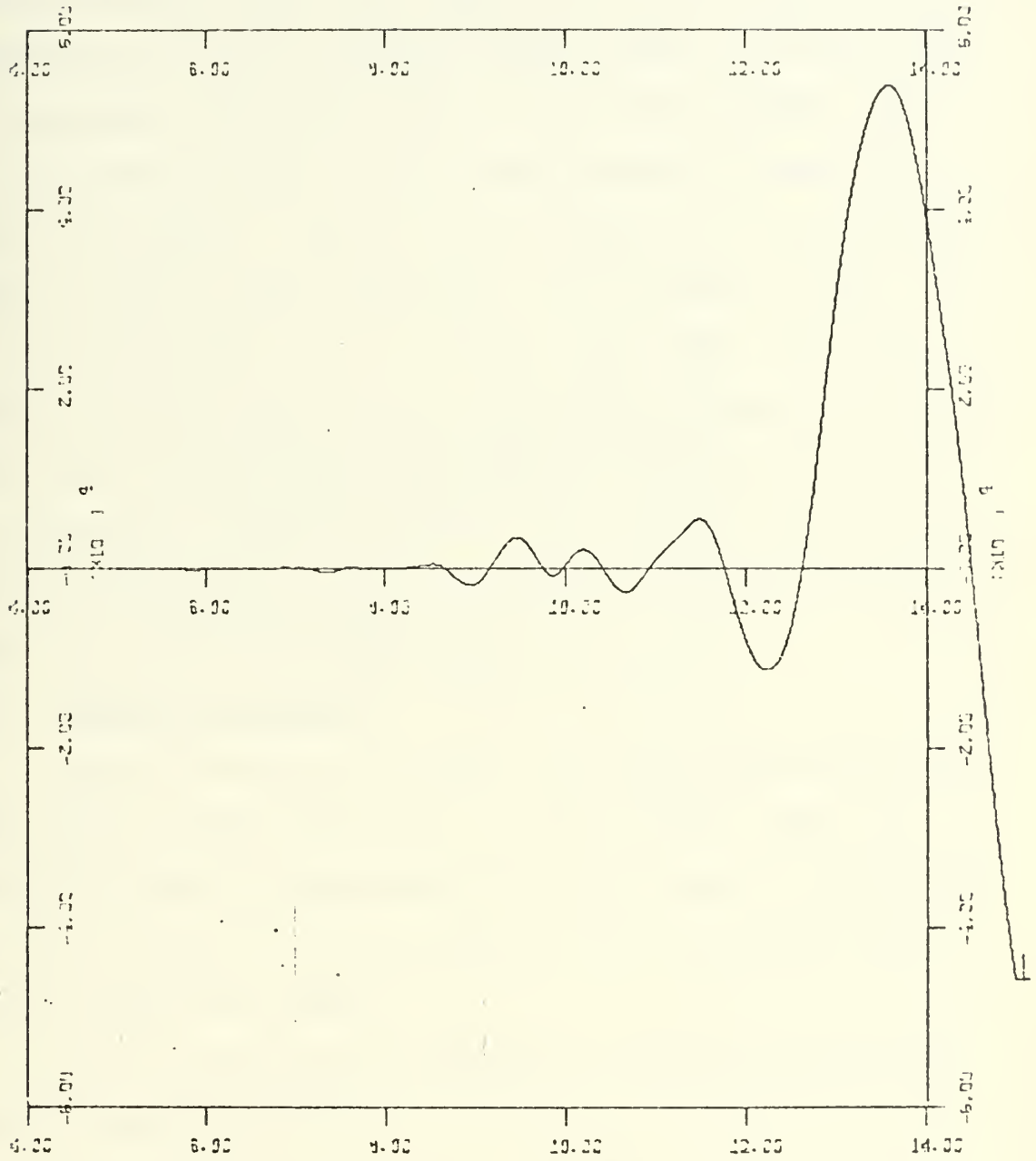
ϕ' vs TIME Showing the Effect of Adding White Noise
(linearly increasing level) Noise Commences at t=0.7 sec.



Xscale: 2.0 sec/inch
Yscale: 5°/inch

FIGURE 3-7

ϕ' vs TIME Showing Loop Instablity due to Saturating the Loop with High level White Noise



Xscale: 2.0sec/inch
Yscale: 2.0×10^{-4}°/inch

Note: Roll loop enters saturation at approximately t=8.0 sec.

necessity of those time consuming integrations during the actual missile simulation, twenty thousand points of 400 hz bandlimited white noise were computed, then written on disk. A subroutine in the DSL program called NOISE was called each time increment. With each call to subroutine NOISE a read was performed from disk and the required white noise was supplied as output. Since a DELT (integration step size) of 0.0001 was used in the simulation, two seconds of white noise were available. At the end of that time, the noise record was rewound and reused. The effect of this repetition of the noise record every two seconds of simulation time on the randomness of the white noise was considered negligible. A sample of the noise used in producing the wander shown in Figure 3-6 is shown in Figure 3-8.

2. Harmonic Generator

Since low amplitude wander vice large amplitude roll wander was of prime importance, it was decided, that to effectively study the phenomenon with the hope of preventing it, a pure harmonic waveform rather than broad band noise should be used as an input.

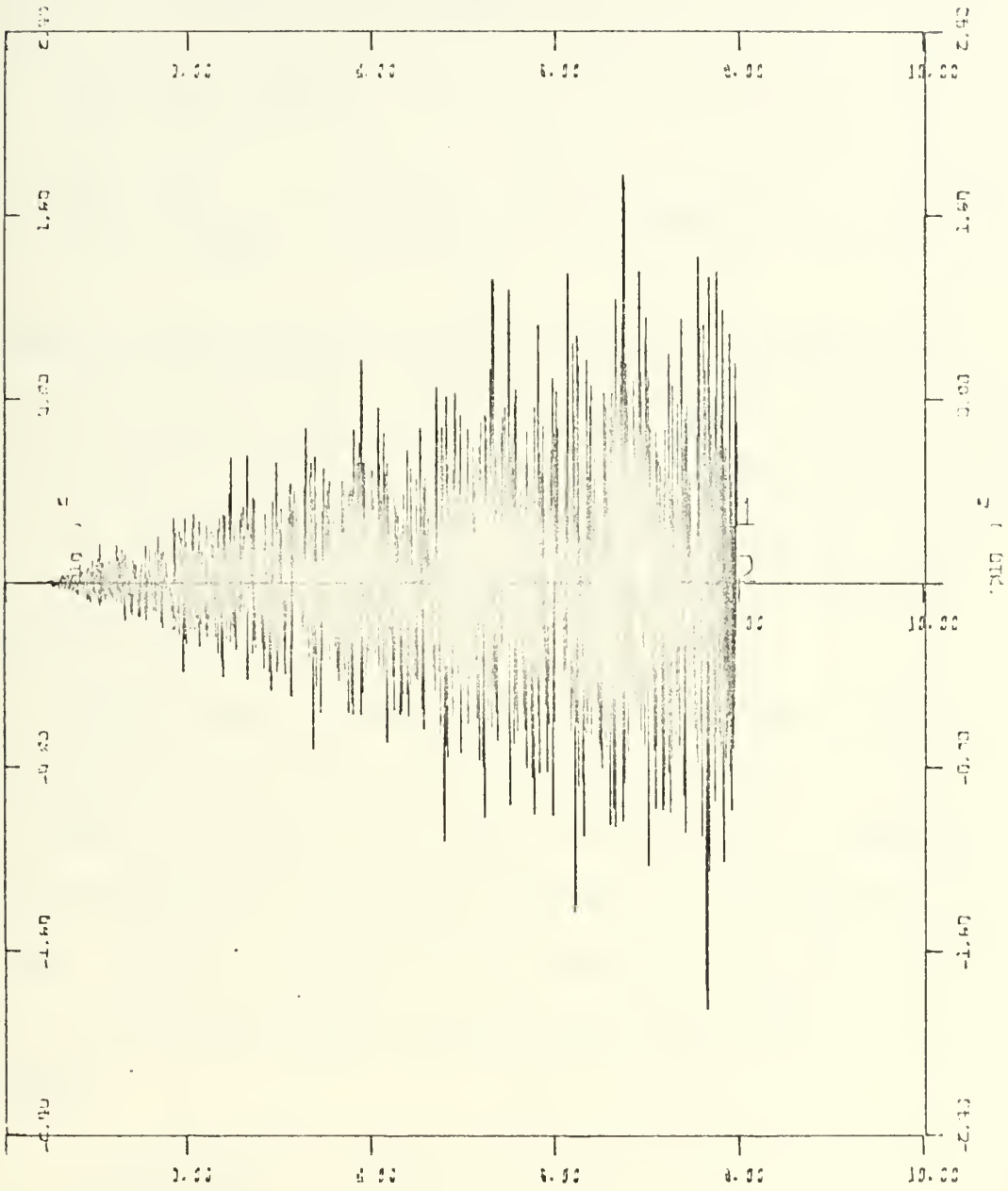
An "in-phase" second harmonic could be generated by using a square law device.

$$HARM = BI^2$$

then if: $BI = BI_{max} \cos \theta$

FIGURE 3-8

White Noise (linearly increasing level)



Xscale: 2.0 sec/inch
Yscale: 80°/sec/inch

$$HARM = BI_{max}^2 \cos^2 \theta$$

$$= BI_{max}^2 \left(\frac{1}{2} + \frac{1}{2} \cos 2\theta\right)$$

$$Bf = BI + HARM$$

$$Bf = BI_{max} \cos + BI_{max}^2 \left(\frac{1}{2} + \frac{1}{2} \cos 2\theta\right) \qquad (15)$$

The harmonic term however carries with it a dc term which would cause a roll offset, and more importantly, might hide the dc error term generated by the control loop. To eliminate this dc level, the output HARM1 was passed through a three stage low pass filter with a -3db frequency of 10 hz and a -18 db/oct roll off.

As discussed earlier in the chapter, when a roll offset exists, a dc error term is produced to null out the loop. When the harmonic is added to the loop, an immediate dc term is produced moving the fundamental waveform off the null axis. Since the harmonic is obtained by squaring the signal in the outer loop, if this signal has such a dc error term then:

$$HARM = BI^2$$

$$= (BI_o + BI_1 \cos \theta)^2$$

$$= BI_o + 2BI_o \cdot BI_1 \cos \theta + BI_1^2 \left(\frac{1}{2} + \frac{1}{2} \cos 2\theta\right) \qquad (16)$$

Equation (16) represents a distorted waveform. As BI_o (the dc component) increases, the effect of the harmonic component decreases. To avoid this problem and again to enable viewing only the wander phenomenon, the waveform used to produce the harmonic was passed through another three stage low pass filter similar to the one described above, (Figure 3-9). Thus at anytime, only a pure sinsoid at a harmonic frequency was introduced.

Addition of this phase locked harmonic is shown in Figure 3-2. The excursion caused by the addition of a harmonic signal producing a signal to noise (harmonic) ratio of approximately +12 db was -11.18°. To produce sustained roll wander, the phase of the harmonic signal had to be shifted rapidly in relation to the fundamental. To accomplish this, a DSL DELAY function block was used. The amount of the delay was varied by setting it equal to:

$$\text{Shift} = \frac{1}{f_o} \sin 2\pi f_s t$$

where:

f_s = 10 hz

f_o = fundamental dither frequency

then: $\text{HARM}'(s) = \text{HARM}(S)\, e^{-(\text{shift})s}$

The resulting waveform is shown in Figure 3-3. The phase shifting frequency used in this simulation was fast enough to cause an increasing amplitude wander to be produced.

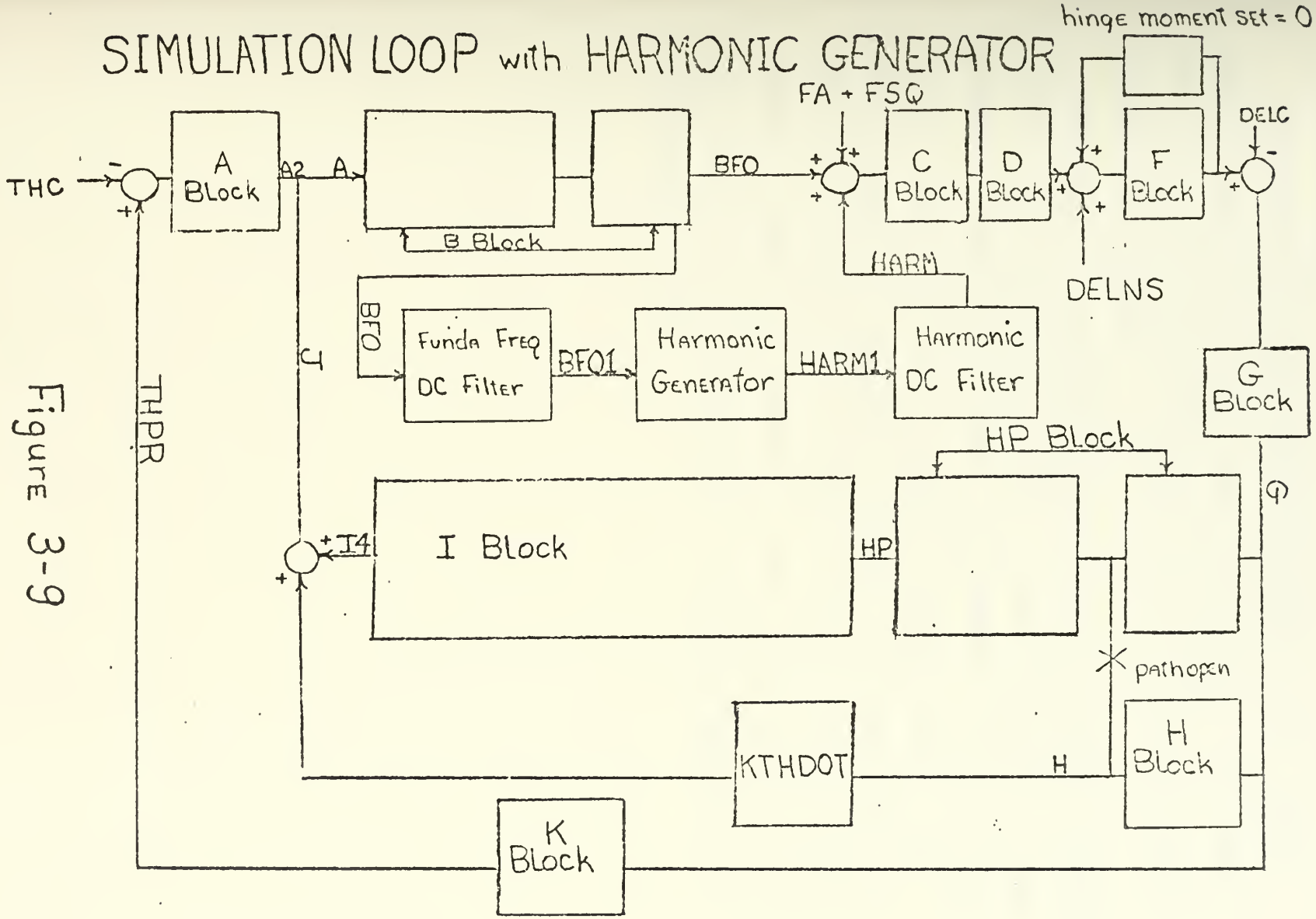

SIMULATION LOOP with HARMONIC GENERATOR
hinge moment set = 0
THC
A Block
A2
A
B Block
BFO
FA + FSQ
C Block
D Block
F Block
DELC
DELNS
HARM
BFO
Funda Freq DC Filter
BFO1
Harmonic Generator
HARM1
Harmonic DC Filter
G Block
HP Block
THPR
I4
I Block
HP
pathopen
KTHDOT
H
H Block
K Block

Figure 3-9

E. CONCLUSIONS

Producing the roll wander with random noise signal ensured that the model was behaving properly. The model reacted to the introduction of noise in the same fashion as the real missile.

The use of the harmonic generator to produce a sustained smooth roll wander proved that the phenomenon was in fact caused by harmonic addition. In addition, the use of the harmonic generator gave valuable insight into the roll loop's response to external stimulus. The fact that the output of the bistable element drives the control surfaces at such a high rate, producing such large missile roll rates, explains why it is so difficult to view an imbalance in the bistable output.

IV. A SOLUTION

A. INTRODUCTION

Once the cause of the roll wander phenomenon had been isolated and verified through testing of the missile model with the noise source and harmonic generator, the problem existed to find a way of preventing this wander. As discussed earlier, two causes of wander due to a noise input exist. Measures to prevent large amplitude wander were not considered. Large amplitude wander would be encountered only under extremely abnormal conditions in which the noise signal totally saturates the control loop. On the other hand, low amplitude wander caused by harmonics present in the noise adding with the fundamental dither frequency, was felt to be a significant problem. In this chapter, the term roll wander will be understood to deal with low amplitude wander.

The most direct method of preventing roll wander would be the attenuation of the dither harmonics in the roll control loop. This was a deceptively difficult problem. There already existed in the control loop several high frequency filters to filter out those high frequency signals generated by the missile's elastic body bending modes. Notch filters vice a simple low pass filter had been utilized to prevent the shifting of the fundamental dither frequency due to the phase shift imposed by the addition of the filters. Inclusion of another set of filters notched at the main harmonic components of the dither fundamental could be used to combat

the frequency addition and resulting roll wander. However, since the dithering action of the roll loop is dependent upon the phase crossover of the loop, which is determined by the loop elements, the frequency of the dither is dependent on the roll loop parameters and varies from missile to missile. More importantly, the dither frequency varies slightly from one set of flight conditions to another. Thus, a notch filter set to filter out harmonics would of necessity have to have a notch wide enough for all frequency variations. This, in turn, would cause an unacceptable decrease in the dither frequency. A self-adaptive notch filter was thus proposed as a possible solution.

B. NOTCH FILTER DESIGN

A filter with conjugate imaginary zeroes located at the frequency of interest, the notch frequency, would provide total attenuation of the signal at that frequency. At frequencies higher than the notch frequency, the filter would have a gain greater than unity. However, the filter must have unity gain at all frequencies other than the notch frequency. To counter the gain of the filter produced by the conjugate imaginary zeroes, a double pole is required. For stability reasons, purely imaginary poles are not possible. The poles must have an attendant ζ term. A problem arises, however, when complex conjugate poles with non zero real parts are introduced to produce a realizable filter.

There is a sign reversal at the notch frequency, which implies a 180° phase shift about the notch. Since the zeroes

are pure imaginaries, having no real parts, there is no phase disturbance at other frequencies due to this shift. This is not the case with the complex poles. The value of ζ in the denominator of the filter expression determines the amount of phase disturbance the filter produces. Frequency components at lower frequencies, specifically at the fundamental frequency, receive a phase shift depending on the magnitude of ζ. The shift in phase produced by the filter changes the phase crossover point of the entire loop. The phase crossover sets the limit cycle oscillation (dither) frequency. The value of ζ also determines the effective width of the filter notch. An engineering trade-off is required, therefore, between the width of the notch and the change produced in the dither frequency.

Finally, a pole must be added at some higher frequency ($\beta = 3000$) to allow the response of the filter to roll off at high frequencies.

The judgment was made in the design of this filter that the zeroes of the filter were to be pure imaginaries. This does not imply that complex conjugate zeroes with non zero real parts would not also provide attenuation at the notch frequency. However, if the zeroes also had non zero real components, the amount of attenuation about the notch frequency would be decreased. With the width of the notch already a critical factor; and since imaginary zeroes are possible (in practice; zeroes with negligible real parts), the filter was designed as stated. Equation (16) is the

form of the transfer function of the filter proposed.

$$H(s) = \frac{s^2 + \omega_o^2}{(s^2 + 2\zeta\omega_o s + \omega_o^2)(s + \beta)} \qquad (17)$$

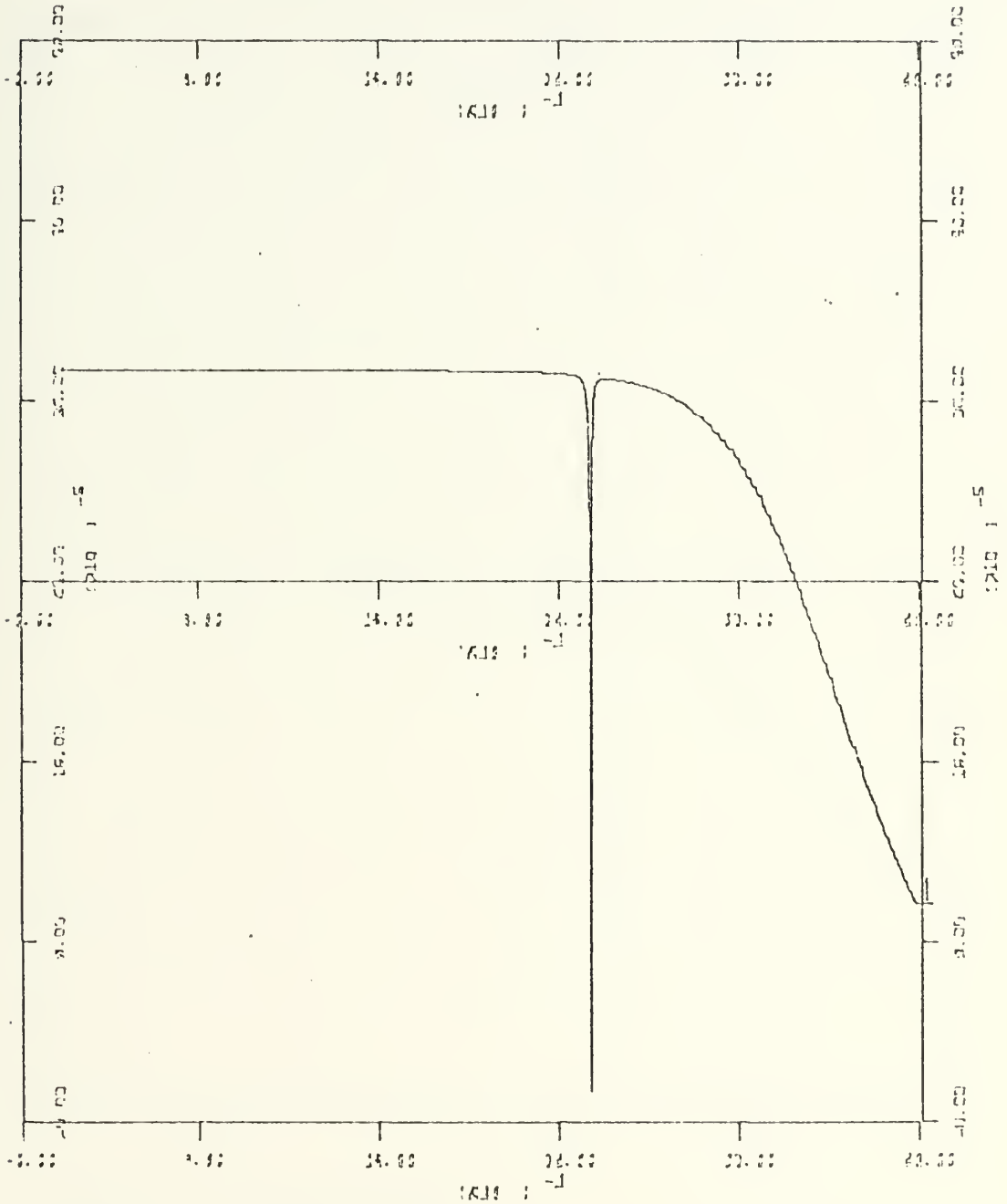
Figures 4-1 through 4-6 show the Bode magnitude and phase plots for the filter's frequency response from 1 hz to 1.59 khz for values of ζ of 0.01, 0.1 and 0.2 respectively. The magnitude plots show the notch formed at the center frequency. (A frequency of 56 hz, twice the fundamental dither frequency was used.) From the phase plots can be seen the amount of phase shift introduced by the filter. With a ζ of 0.2, a phase shift of -25° was experienced at 28 hz. For ζ equal to 0.1, the phase shift was -17.7°; and with ζ equal to 0.01, the phase shift was -10.8°.

The filter was tested outside of the missile model. It was found that adequate attenuation of a signal having a bandwidth of 0.2 hz centered at the filter notch frequency was not obtained with values of ζ less than 0.07. When the filter was tested in the missile model with ζ set to 0.01, the frequency of the dither was reduced from 28.75 hz to 26.3 hz; with ζ set to 0.1, the frequency of the dither dropped to 24.5 hz, and with a ζ of 0.2, the frequency dropped to approximately 22 hz.

It was decided to use a value for ζ of 0.1 for minimum disturbance in the loop with maximum attenuation of the harmonic frequencies. Two filters were inserted, in cascade,

FIGURE 4-1

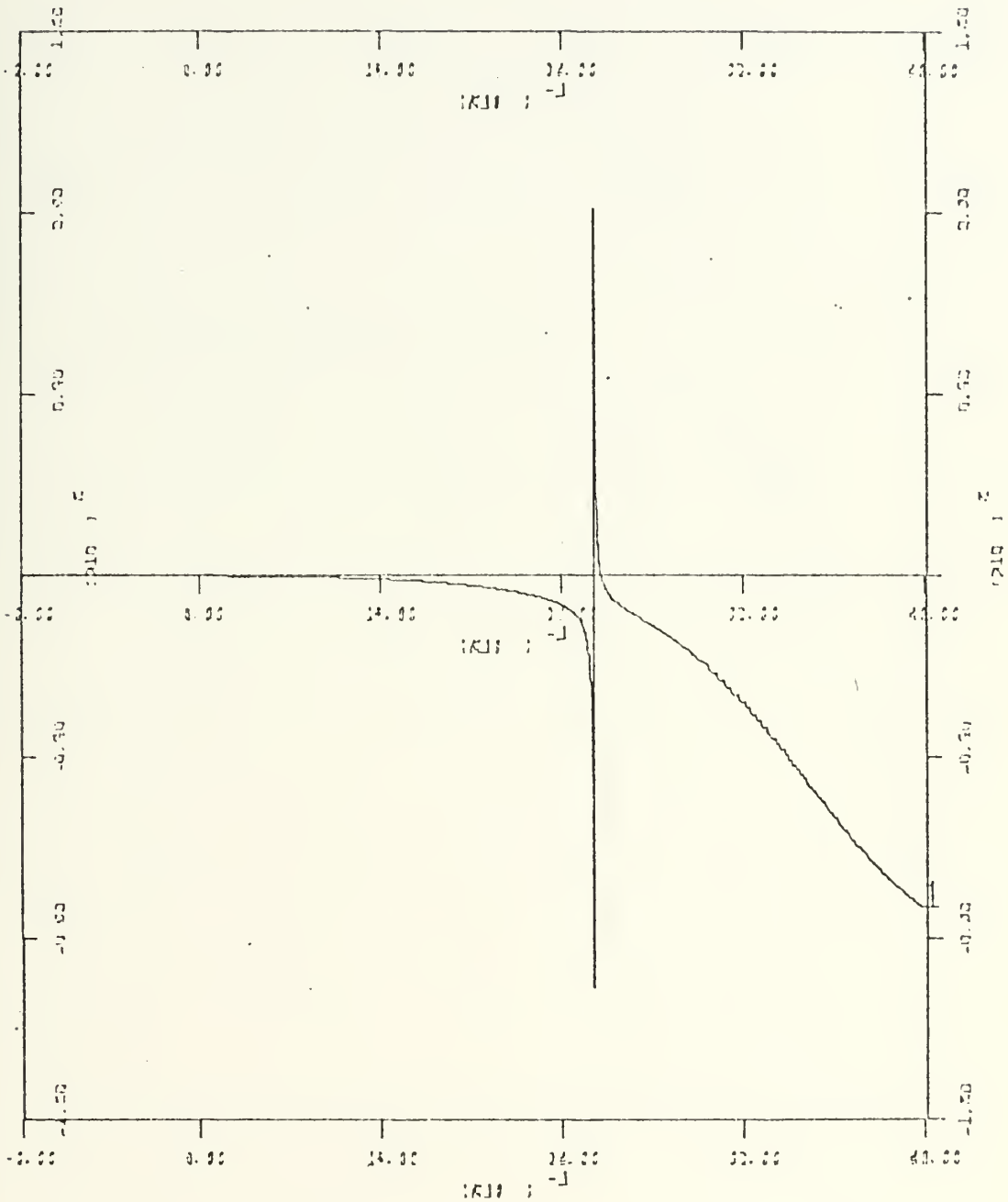
Bode Magnitude Plot of the Notch Filter with ζ=0.01



Xscale: log(10hz)/inch
Yscale: 8.0×10^{-5} units/inch

FIGURE 4-2

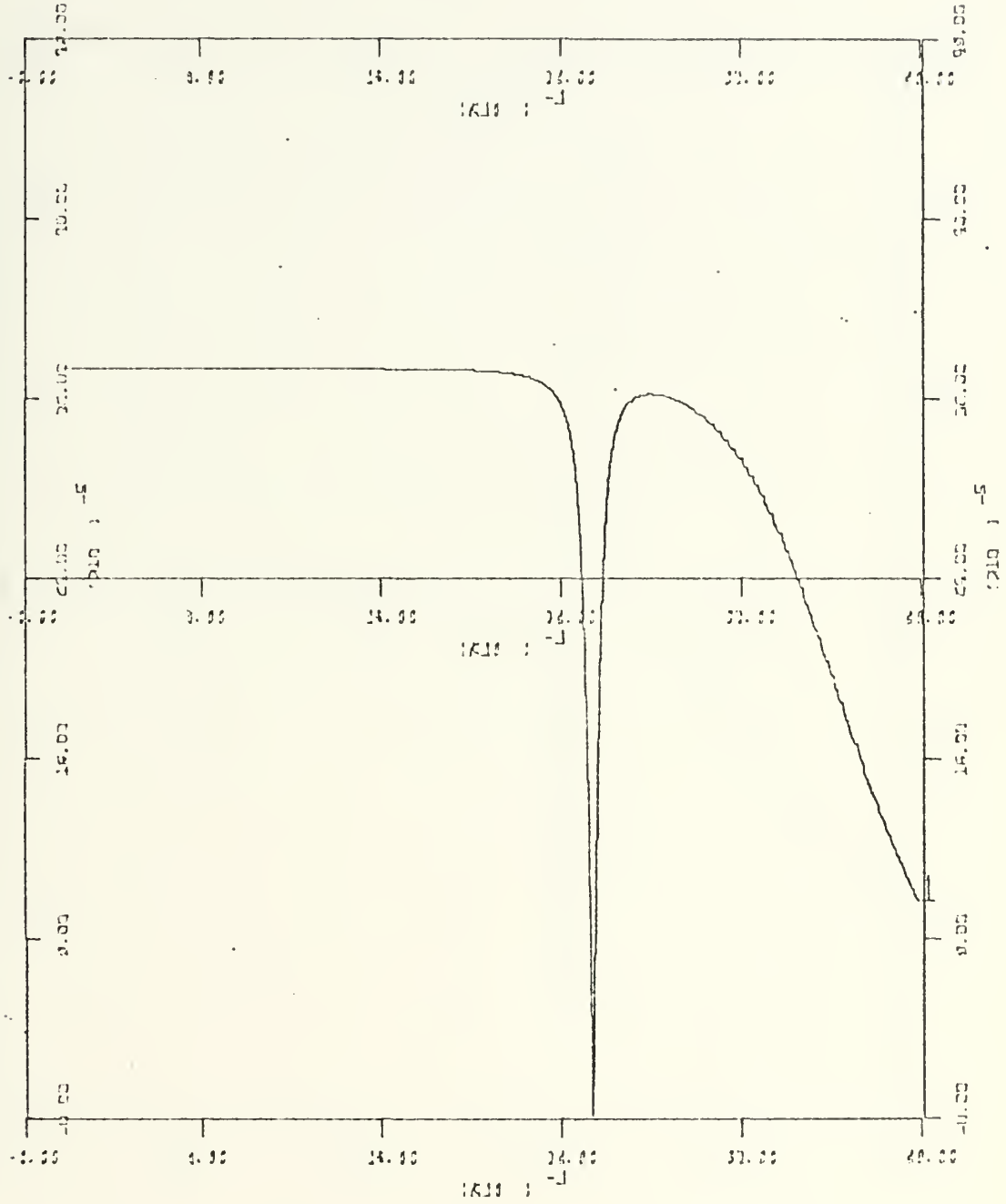
Bode Phase Plot of the Notch Filter with ζ=0.01



Xscale: log(10hz)/inch
Yscale: 40°/inch

FIGURE 4-3

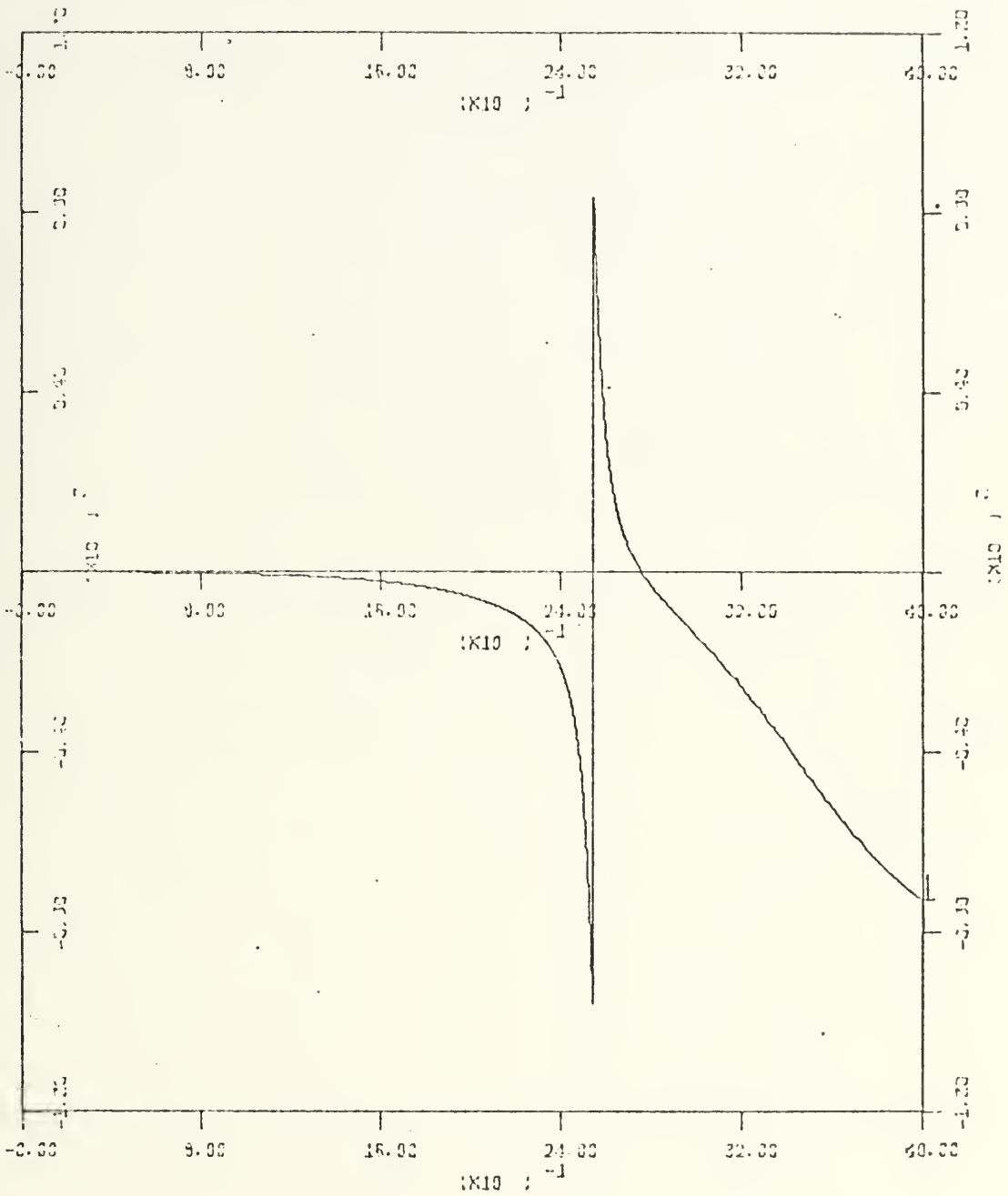
Bode Magnitude Plot of the Notch Filter with $\zeta=0.1$



Xscale: log(10hz)/inch
Yscale: 8.0×10^{-5} units/inch

FIGURE 4-4

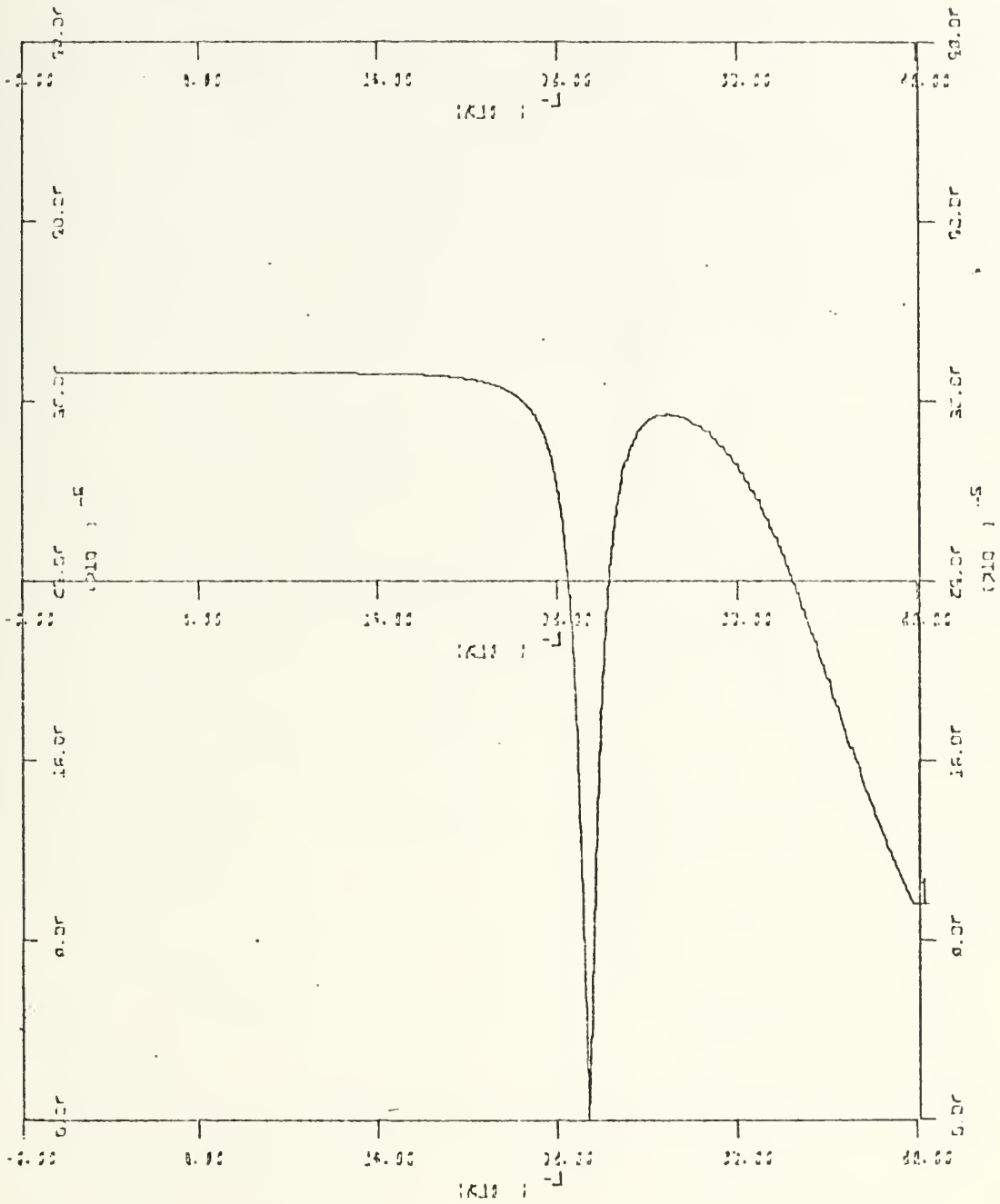
Bode Phase Plot of the Notch Filter with ζ=0.1



Xscale: log(10hz)/inch
Yscale: 40°/inch

FIGURE 4-5

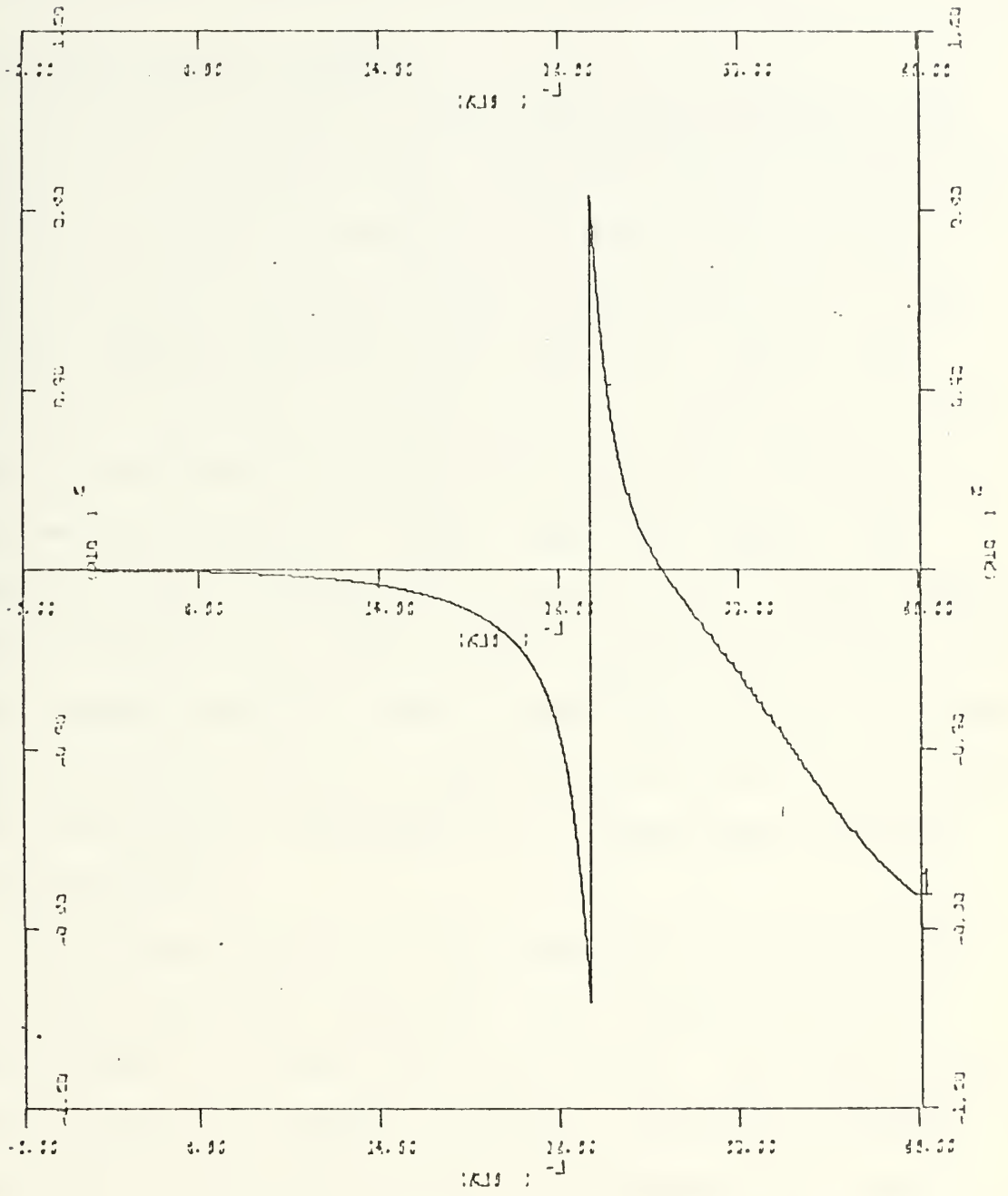
Bode Magnitude Plot of the Notch Filter with ζ=0.2



Xscale: log(10hz)/inch
Yscale: 8.0×10^{-5} units/inch

FIGURE 4-6

Bode Phase Plot of the Notch Filter with ζ=0.2



Xscale: log(10hz)/inch
Yscale: 40°/inch

with notch frequencies set to filter out the second and third harmonic frequencies of the fundamental.

C. SELF ADAPTIVE FEATURE

The notch frequency of the filter is specified by the value of ω_o in Equation (16). The compromise concerning the value of ζ, which determines the width of the notch was based upon the assumption that ω_o would be within ±0.1 hz of the harmonic frequency to be filtered. However, as previously discussed, several factors, including the inclusion of the filter itself into the loop, will cause the fundamental frequency to vary. When this happens, the harmonic of the fundamental also varies, by a larger amount, proportionate to the order of the harmonic in question. If the harmonic component moves in frequency outside the .2 hz wide notch, the filter no longer has any effect. The filter must therefore be self-adaptive; it must have a variable notch frequency.

The filter obtains its self-adaptive feature by first measuring the dither frequency, then setting the filter coefficients to produce the correct notch. The filter produces an initial transient response. Constantly changing the filter coefficients, which is effectively the same as constantly inserting a different filter, disturbs the control loop. This disturbance is seen as a random wander in the position loop. To minimize this effect, the filter coefficients must be held constant or be required to change slowly

in a smooth fashion. This will only be the case if the dither remains fairly stable.

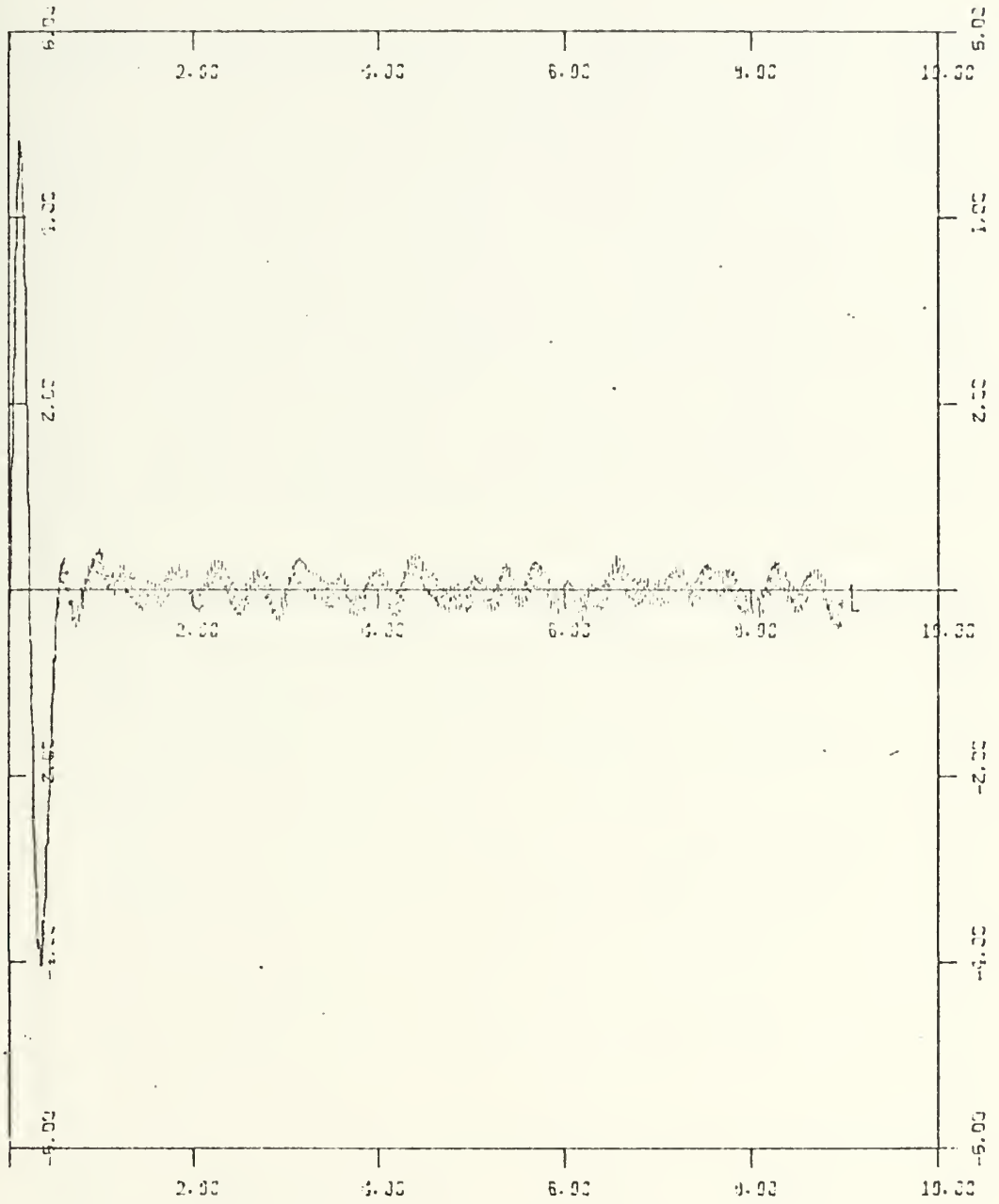
A jitter in the dither frequency is produced in the simulation because continuous integrations are being simulated by discrete numerical operations. Also, since the output of the bistable element is a pure square wave, instantaneous frequency was most easily measured at this point. Any external influence which disturbs the relative sizes of the states of the bistable element would also produce a fluctuation in the instantaneously measured frequency. The measured value of instantaneous frequency did not therefore remain constant. When the filter coefficients were changed each integration interval, to correspond to this instantaneous frequency, a wander with a frequency of approximately 2.5 hz was observed. The filter was itself causing a roll wander phenomenon (Figure 4-7).

This side effect was significantly reduced by using an average value for the frequency. The frequency was measured each integration interval, then averaged over six cycles. As each new cycle was recorded, it was then averaged with the previous five cycles; thereby "fairing" out any frequency jitter. The result is seen in Figure 4-8.

It is obvious that in actual practice the construction of such a self-adaptive filter is not quite so simple as the simulation of one. However, techniques do exist for the production of such filters, either analog or digital. This

FIGURE 4-7

ϕ' vs TIME with Self Adaptive Filter in Control Loop (no input)
Adaptation using Instaneous Frequency

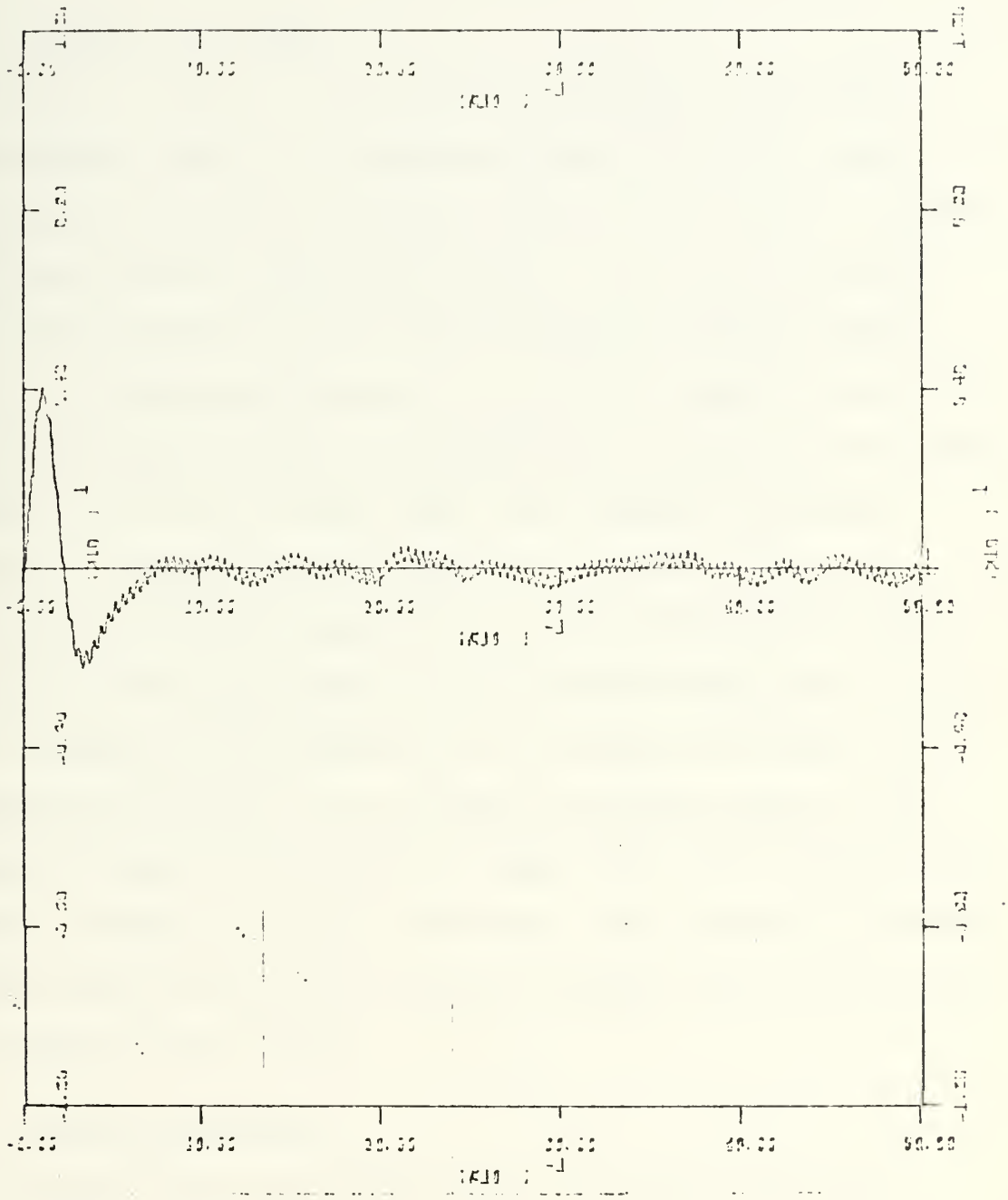


Xscale: 2.0 sec/inch
Yscale: 2.0°/inch

Note: Initial excursion is due to the initial conditions used in the simulation.
Note: Dither frequency is measured each integration interval. Filter coefficients are updated at the same rate.

FIGURE 4-8

ϕ' vs TIME with Self Adaptive Filter in Control (no input) Adaptation using Averaged Frequency



Xscale: 1.0 sec/inch
Yscale: 4.0°/inch

Note: Dither frequency is measured and averaged over 6 cycles. Filter coefficients are updated using averaged value of frequency.

thesis concerned itself only with the filter's effect on the control loop, and not on filter construction.

D. TESTING THE FILTER

Since it was the effect of the addition of harmonic frequencies with the fundamental dither frequency which produced the distorted waveform responsible for roll wander, both the harmonic generator and the filter were inserted into the loop just prior to the bistable element (Figure 4-9). In the actual missile, it would be expected that such a filter would be placed at the output of the roll rate gyro; since it is the roll rate gyro which inputs the actual harmonic frequencies into the loop.

With the filter installed, the control loop model was tested in the same fashion as described in Sections II and III. Figure 4-8 and Figures 4-10 through 4-14 show the response of the model to no input, step input and a tail induced torque, respectively. These runs were all made with normal, identical flight conditions. The missile's response during high altitudes and speeds remained the same as before inclusion of the filter.

E. WANDER SUPPRESSION

The runs just described showed that with the self-adaptive filter in the loop, the model still responded properly to input commands. The model was then tested again with the harmonic generator. Figure 4-15 and Figure 4-16 clearly show the filter's effect. The roll wander is totally

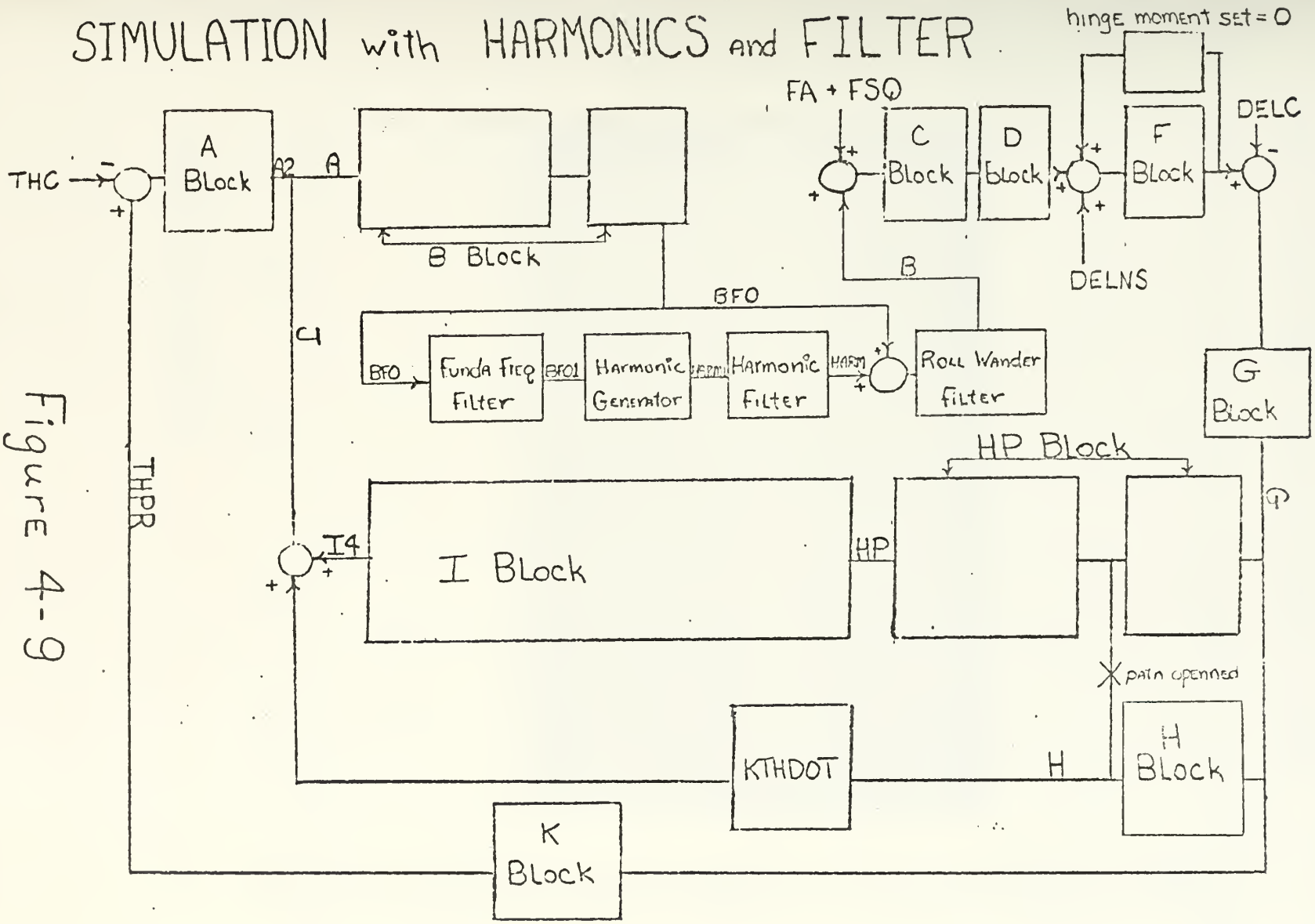
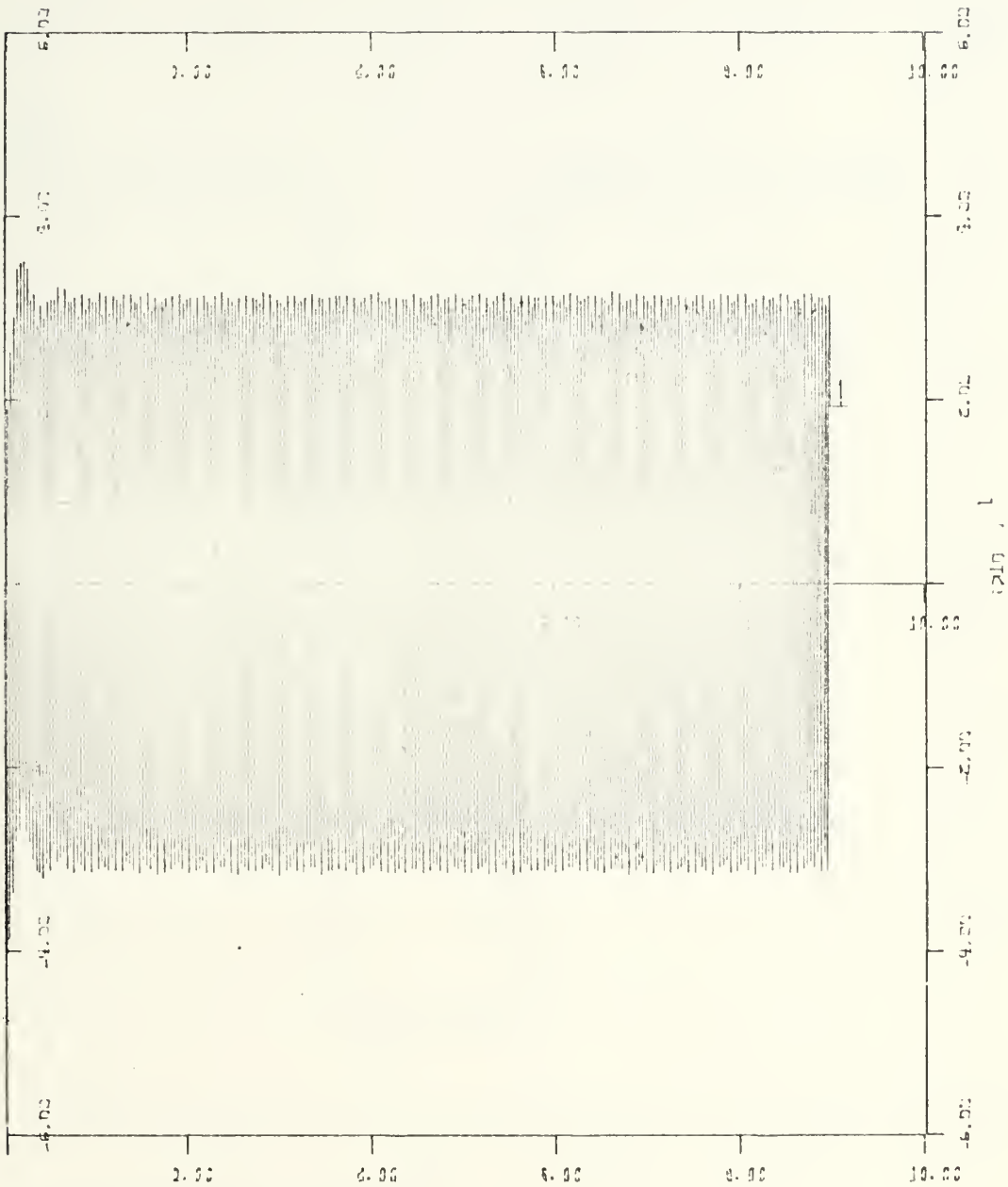

SIMULATION with HARMONICS and FILTER
hinge moment set = 0
FA + FSO
THC
A Block
A2
A
B Block
C Block
D block
F Block
DELC
DELNS
B
BFO
C1
BFO
Funda Freq Filter
BFO1
Harmonic Generator
HARM
Harmonic Filter
HARM
Roll Wander Filter
G Block
HP Block
THPR
I4
I Block
HP
Q
pitch openned
KTHDOT
H
H Block
K Block

Figure 4-9

FIGURE 4-10

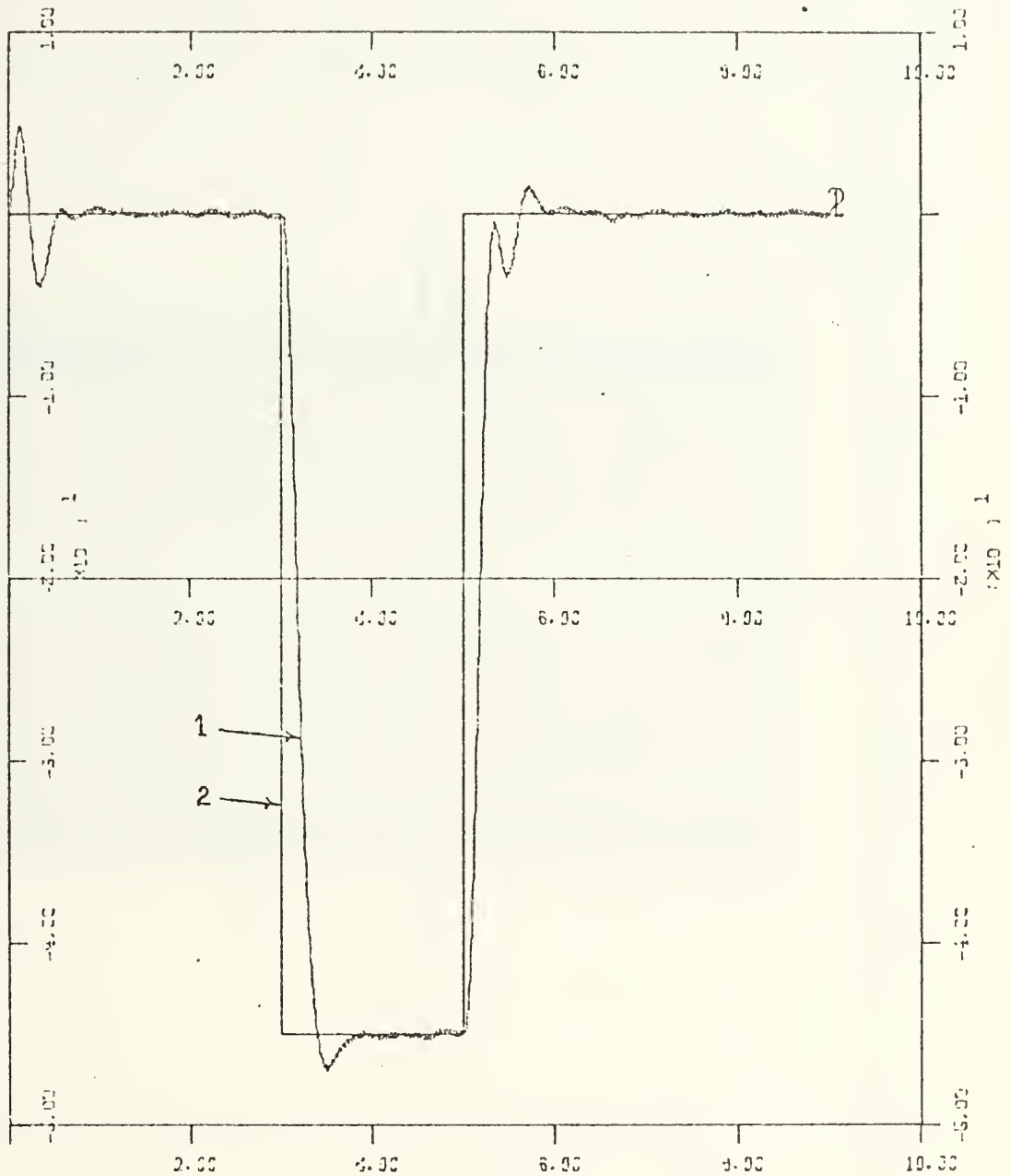
$K_\delta F_\delta$ vs TIME with Filter in Loop (no input)



Xscale: 2.0 sec/inch
Yscale: 20°/sec/inch

FIGURE 4-11

ϕ' vs TIME with Filter in Loop and -45° step input at t=3.0 sec

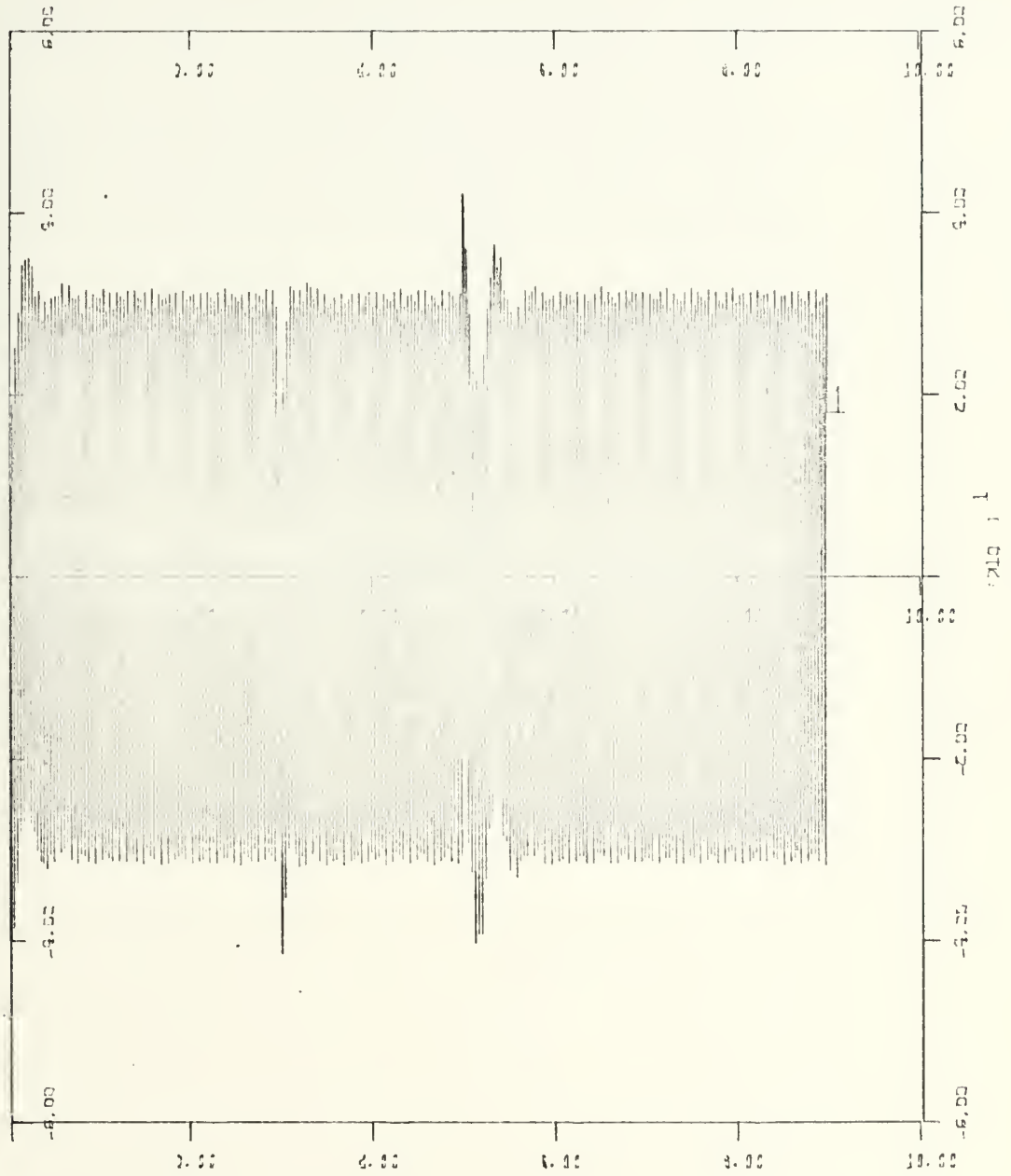


Xscale: 2.0 sec/inch
Yscale: 10°/inch

Curve #1 ϕ'
Curve #2 Step Input

FIGURE 4-12

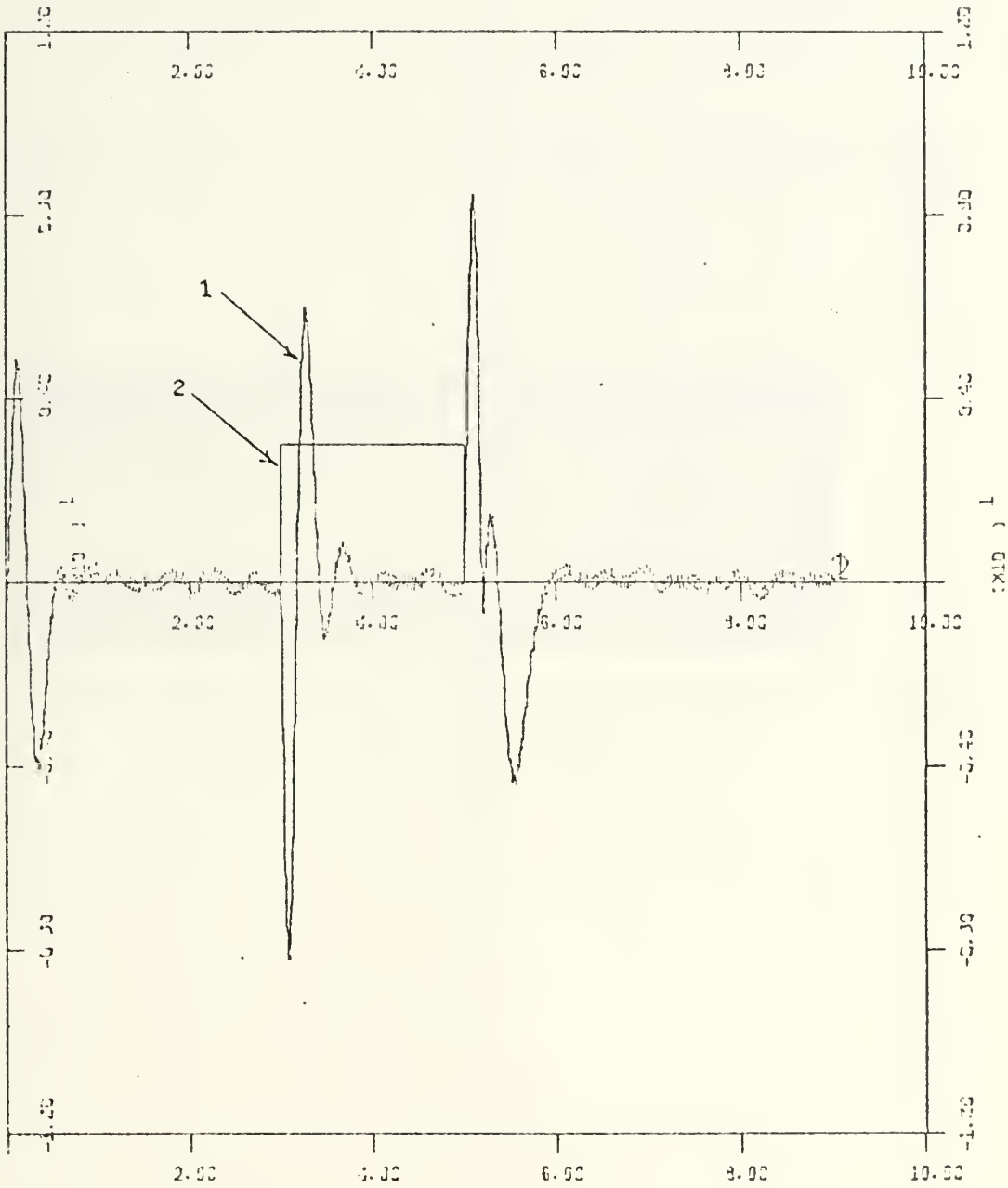
$K_{\delta} F_{\hat{\delta}}$ vs TIME with Filter in Loop and a -45° step input at t=3.0 sec.



Xscale: 2.0 sec/inch
Yscale: 20°/sec/inch

FIGURE 4-13

ϕ' vs TIME with Filter in the Loop and a +3° step torque (δ_c) at t=3.0 sec

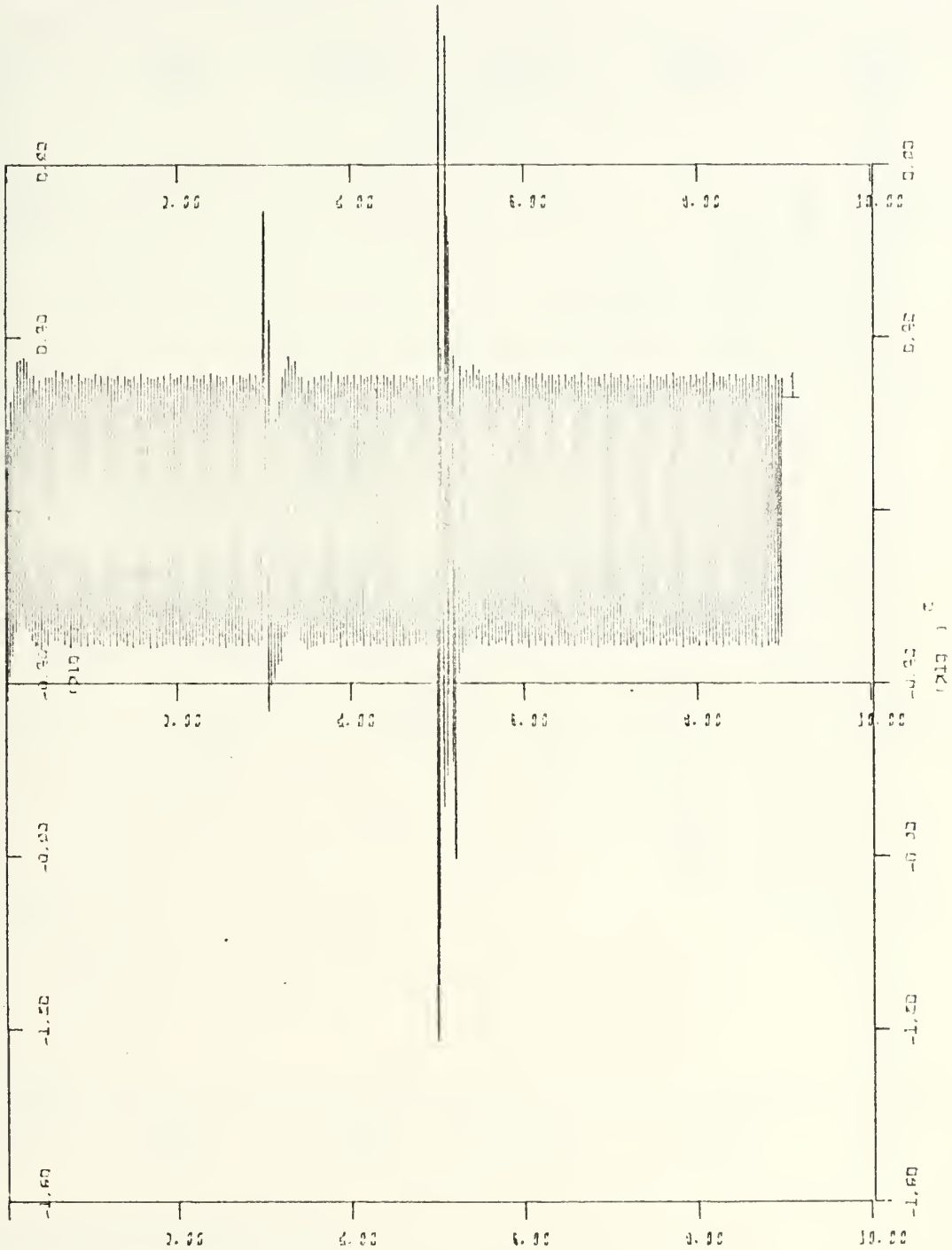


Xscale: 2.0 sec/inch
Yscale: 4°/inch

Curve #1 ϕ'
Curve #2 δ_c

FIGURE 4-14

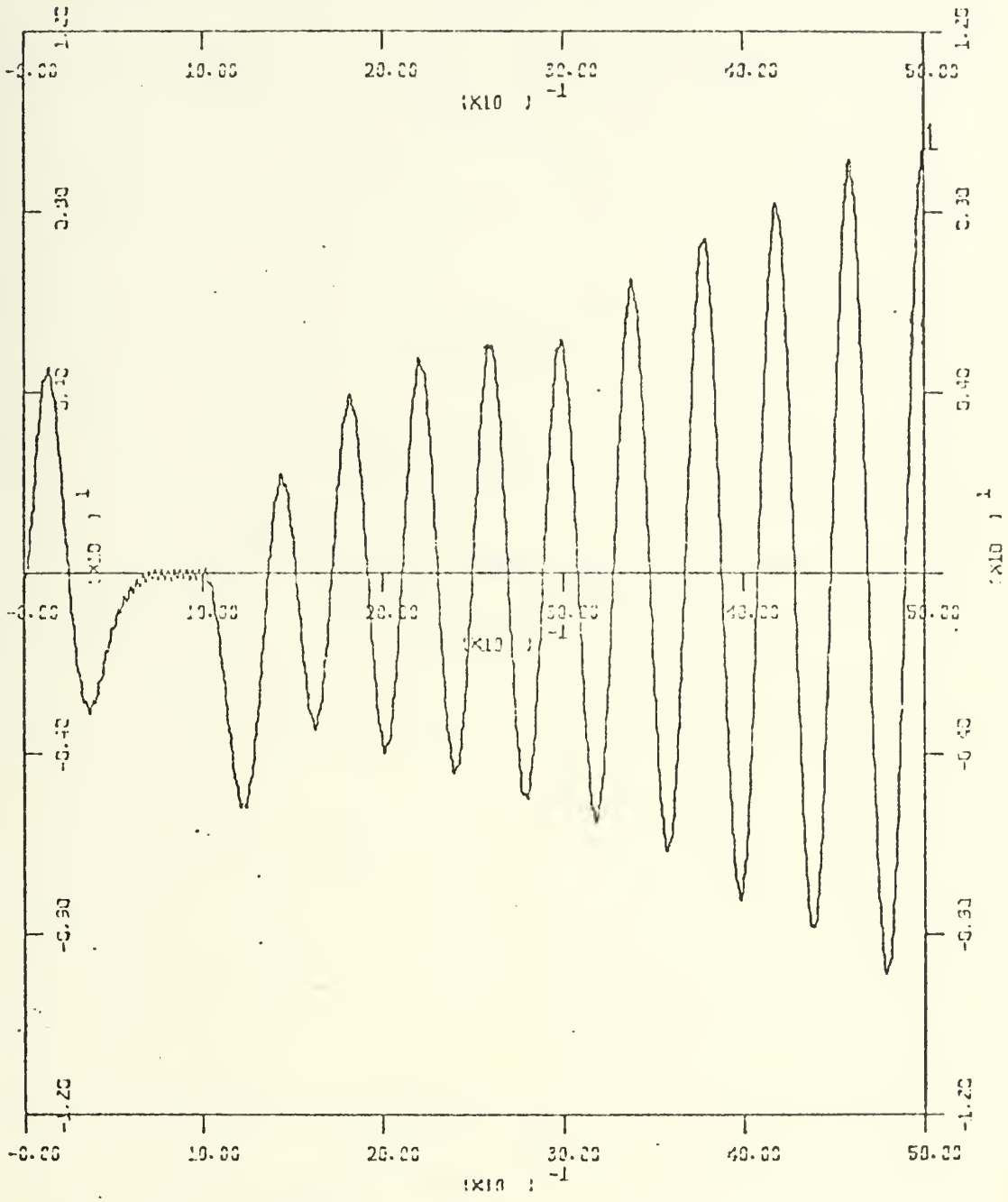
$K_{\delta}F_{\delta}$ vs TIME with Filter in the Loop and a +3° step torque(δ_c) at t=3.0 sec.



Xscale: 2.0 sec/inch
Yscale: 40°/sec/inch

FIGURE 4-15

ϕ' vs TIME Showing the Effect of a Varying Phase Harmonic
(Harmonic Commences at t=1.0 sec)

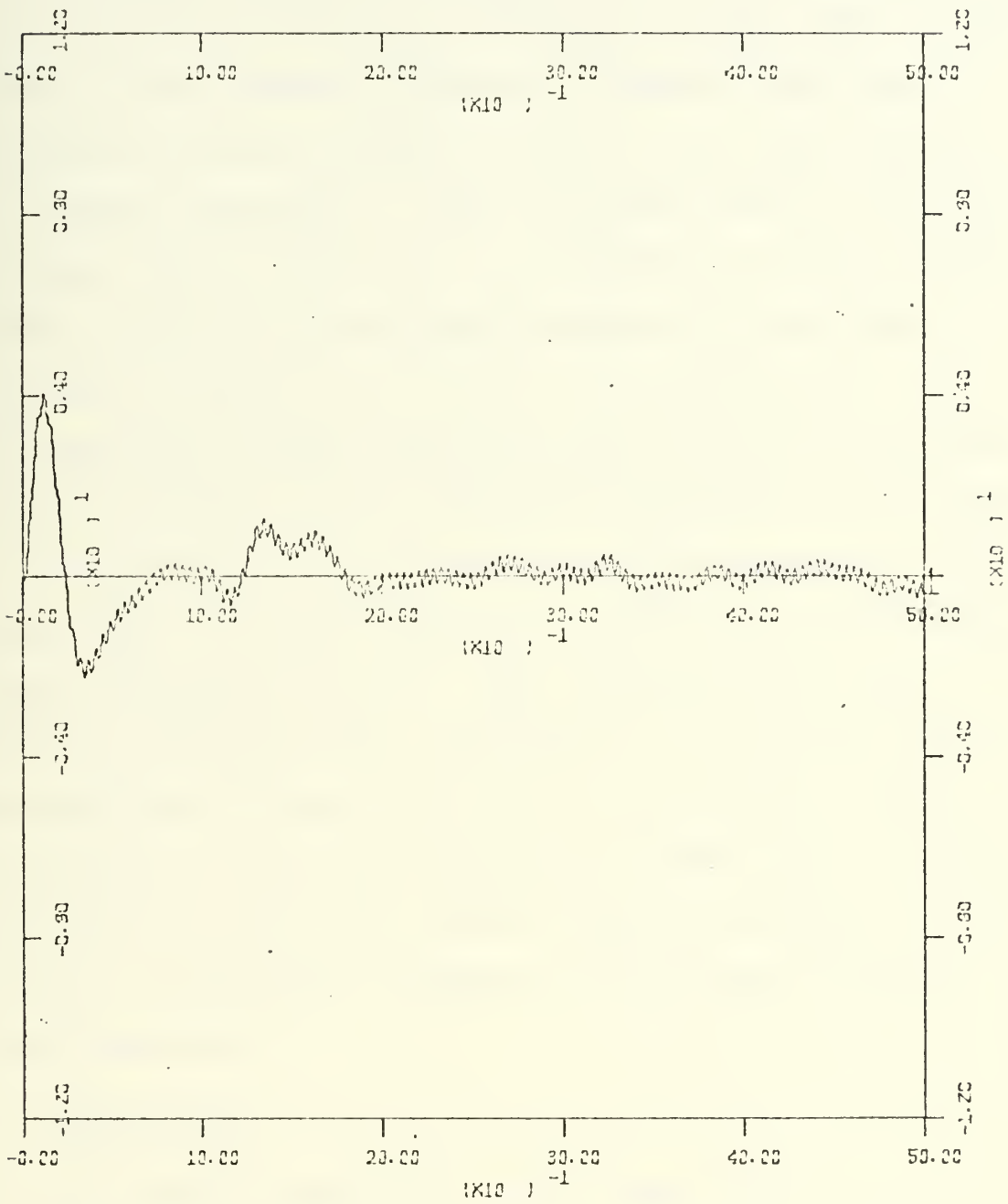


Xscale: 1.0 sec/inch
Yscale: 4°/inch

Note: Initial excursion is due to the initial conditions used in the simulation.

FIGURE 4-16

ϕ' vs TIME with Filter in Loop and with a Varying Phase Harmonic (Harmonic Commences at t=1.0 sec.)



Xscale: 1.0 sec/inch
Yscale: 4°/inch

Note: Initial excursion is due to the initial conditions used in the simulation.
Note: The excursion at t=1.0 sec. is due to the harmonic causing a momentary shifting of the dither frequency outside the notch bandwidth.

suppressed by the notch filter. The transient disturbance seen in Figure 4-16 is caused by the initial "switching in" of the harmonic signal at t equals one second. The initial effect of the harmonic produces a change in the dither frequency, with an attendant change in the output signal of the harmonic generator. Since the notch filter is using an averaged value of frequency in computing the filter coefficients, the shift in harmonic frequency is momentarily outside the range of the notch. Roll wander starts. As the averaged frequency approaches the actual loop frequency the notch shifts and the wander is suppressed by the filter.

Figures 4-17 and 4-18 show the model's response to the noise input. A noise signal will disturb the roll loop and cause the missile to roll. This, however, is not roll wander. Comparing these two figures, it can be seen that the amplitude of the wander has been decreased by about 50 per cent. The rougher appearance of the curve in Figure 4-18 is due to the fact that the random disturbance of the noise is becoming more dominant as the harmonic addition is being suppressed.

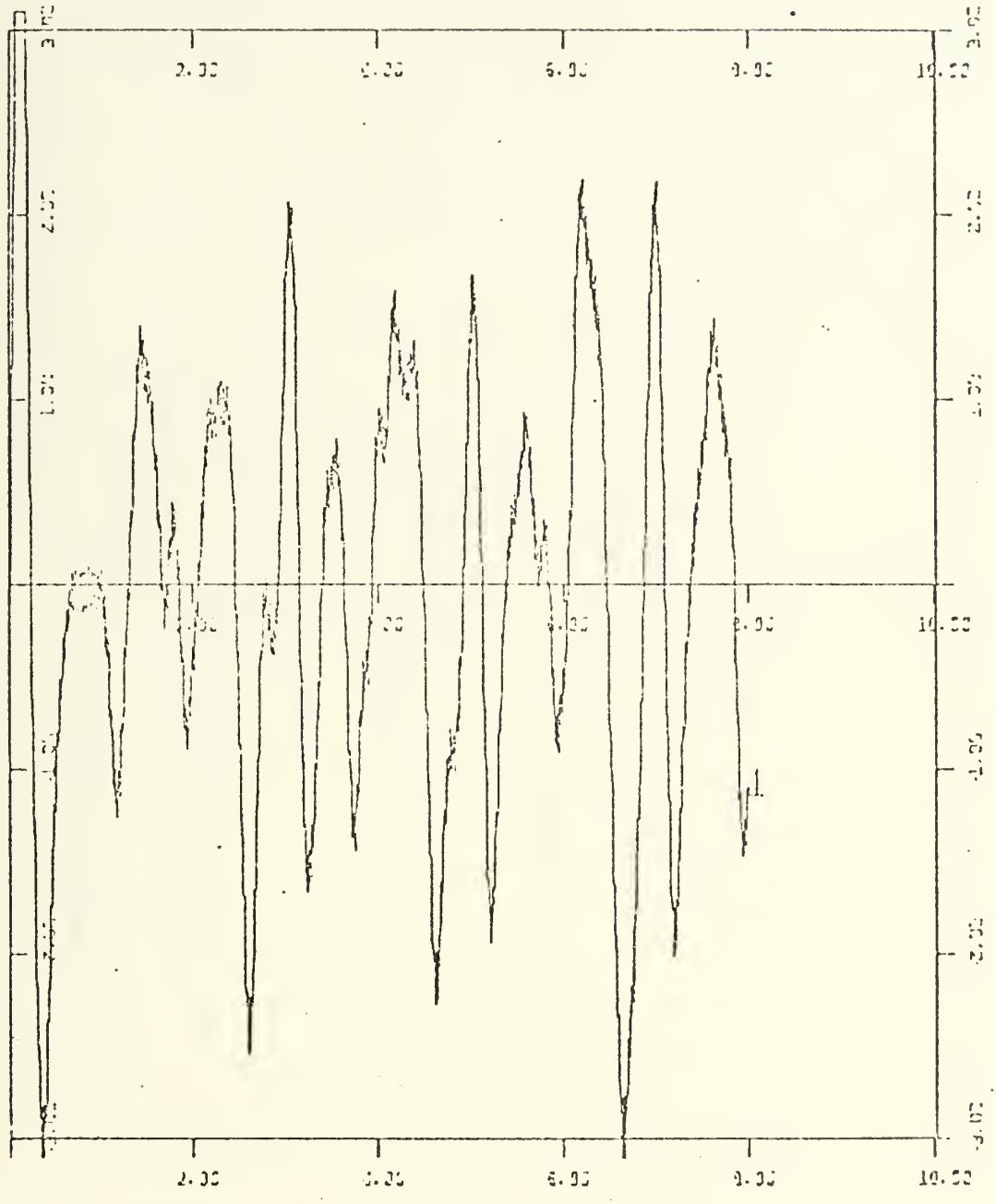
F. CONCLUSIONS

The primary objectives of this thesis were:

1. Construct an accurate digital model of the missile roll control loop.
2. Produce in the model the phenomenon of roll wander with an explanation of the sources and mechanisms of the wander.
3. To produce a means of preventing or at least significantly reducing the occurrence of this phenomenon.

FIGURE 4-17

ϕ' vs TIME Showing the Effect of Adding White Noise (constant level). Noise Commences at t=0.7 sec.

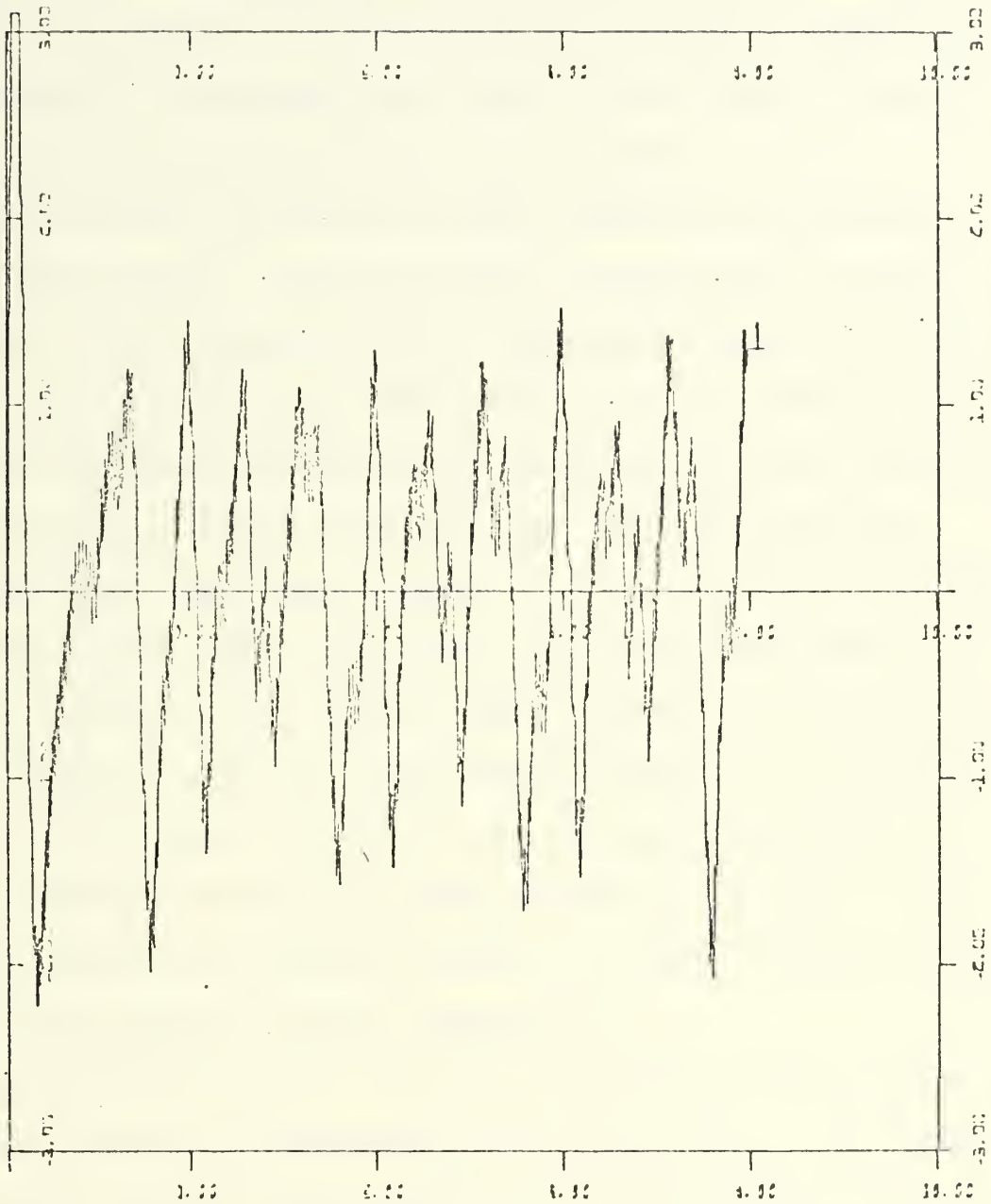


Xscale: 2.0 sec/inch
Yscale: 1°/inch

Note: Initial excursion is due to the initial conditions used in the simulation.

FIGURE 4-18

ϕ' vs TIME with Filter in Loop and White Noise Added
Noise Commences at t=0.7 sec.



Xscale: 2.0 sec/inch
Yscale: 1°/inch

The first two objectives have been achieved in this thesis. As shown in Sections I, II, and III, the model behaves in the same fashion as the real missile. The digital simulation of the missile now exists at the Naval Postgraduate School, and is available for further study.

The solution presented in this chapter as a cure to the roll wander problem works and will prevent this phenomenon. However, a self-adaptive filter has several weaknesses which limit its effectiveness. The inclusion of the filter into the control roll has reduced the fundamental dither frequency significantly. Such a reduction may cause interference problems with the missile guidance system. It was found that changing the filter coefficients, to change the notch frequency, disturbed the control loop, causing a fluctuation in the fundamental frequency and random wander in the position loop. To minimize this effect, an averaged frequency was used. Because of this, the filter did not respond to frequency changes as quickly; and the roll wander caused by the noise input was not totally suppressed.

It is believed that much room exists for future thesis work in the area of preventing the roll wander. The largest portion of the work involved in this thesis was the construction of an accurate model and the production of the roll wander. With the model now in existence, future thesis students can use it and concentrate on the development of a positive efficient system to prevent roll wander.

V. RECOMMENDATIONS FOR FUTURE STUDY

As discussed in the previous chapter, continued study is needed in the implementation of an adaptive filter for the suppression of roll wander. The filter used in this thesis produced a decrease in dither frequency of almost 4 hz. This alone may make the filter acceptable.

Another area of study, apart from the study of the missile control system, also became evident during the course of this thesis. That is the study and the use of digital simulation languages. As is often the case with any research, the preparation consumed as much time as the investigation itself. I am referring to the production of the DSL plot package. DSL lay dormant at the Naval Postgraduate School because of its inability to output to the CALCOMP plotter. Once an operational plot package was produced, the value and the power of DSL as a simulation tool rapidly became apparent.

IBM's Continuous System Modeling Program (CSMP), another simulation language at this school, also at present requires considerable programming to access the CALCOMP plotter. It would require only slight modification to DSLPLOT and minor procedural changes to the system control cards for CSMP to make CSMP output to the plotter via DSLPLOT.

It must be remembered that as with any simulation routine DSL is time consuming; and on a time-shared system such as the IBM 360/67 System used at the Postgraduate School, turn around time is, at best, marginal. The IBM 360/67 System

has interactive graphics terminals. If DSL were modified via appending FORTRAN programs to output and input via a graphics terminal, a pseudo real time system could be produced. The advantages of such a system would be great. The student could modify his system, and automatically see the result. If the program misbehaved, the student could terminate the run immediately, rather than waiting the full specified time of the run and the time required to exit the system prior to being able to study the program and take corrective action. Such a system could easily allow the user to accomplish in a two or three hour session what might require days or weeks to accomplish off line. The modification of DSL to produce such a system would be extensive but rewarding.

The other modeling languages on file at the Postgraduate School (CSMP, ECAP, LISA, etc.) all would benefit immensely by similar modifications. The saving in systems time by removing such jobs from the normal job queues and rescheduling them during hours of minimum load would also be significant.

The emphasis in control engineering has shifted dramatically to the utilization of the digital computer, both for problem solving and system modeling. However, the structure and implementation of the simulation programs available is not explicitly taught. A course built around system modeling using such programs would be highly beneficial.

APPENDIX A

A. INTRODUCTION

Digital Simulation Language (DSL) is an IBM System/360 FORTRAN IV program for the digital simulation of continuous system dynamics. Its nonprocedural problem-oriented input language accepts problems expressed either at the analog block diagram level or as systems of ordinary differential equations. Since the input language is nonprocedural, proper statement sequence is established by an internal sort based on correct information flow. DSL includes FORTRAN as a subset, thereby extending its power to handle nonlinear and time-variant problems of considerable complexity. A translator converts the input statements into a FORTRAN subprogram, which is compiled and executed to accomplish the simulation.

A significant feature of DSL is centralized integration, which means that all integrator outputs are computed simultaneously at the end of each calculation interval. The user may specify one of several integration methods, including fifth order Milne Predictor and Corrector and second and fourth order Runga Kutta.

DSL as originally presented at the Naval Postgraduate School, outputs to the high speed printer only, either as printed output or printer plots. Due to software incompatibilities, DSL as initially installed would not output to the CALCOMP XY plotter. This thesis, as with many simulation studies, required the CALCOMP as an output. For this reason,

and since DSL had so many advantages, extensive work was done prior to constructing the missile simulation on a FORTRAN plotting program to output the DSL program via the CALCOMP plotter. This program, DSLPLOT and its associated FORTRAN program PLOTHALL, will be discussed later.

B. PROGRAM FLOW

At the start of the program, all program variables are set to the initial conditions given. TIME, the independent variable, is set to zero, and a run counter KSIM is incremented if it is the beginning of a new parameter study. Initially KSIM is set equal to 1. At this point, the region of the program called the INITIAL Region is entered. In the INITIAL Region, arithmetic operations and logical statements to be executed only at the beginning of a particular run are computed. An integer flag KEEP, is set equal to zero, and the DERIVATIVE Region is called.

The DERIVATIVE Region represents that portion of the simulation which involves integration and the calculation of the derivatives. The basic interval in the independent variable for each pass through this region is the calculation interval DELT. It is for this reason that the DERIVATIVE Region should include all time (independent variable) dependent calculations which must be performed at the same frequency required by the selected integration algorithm to assure accurate numerical integration.

The nonprocedural nature of the input language is the result of the DERIVATIVE Region. The DERIVATIVE Region is the only region in which an internal sort routine is utilized to establish the correct statement sequence. Only structure statements are sorted and sequenced. A statement is properly sequenced if all its inputs are available either as input parameters or initial conditions or as previously computed values in the current calculation interval. Outputs of INTGRL and MEMORY blocks are known because their initial values are given. The internal sequencing procedure begins first with integrator outputs, then with outputs of MEMORY blocks and finally with those remaining output variables in the DERIVATIVE Region that have not been included in the sort sequences of integrators and MEMORY blocks. It is important to realize that the sorting algorithm is unable to sequence the statements properly unless it can assume a known output variable as a starting point of a sort sequence. As previously stated, this output is provided by either an INTGRL or MEMORY block. A loop which does not contain at least one such block is called an IMPLICIT or ALGEBRAIC loop and cannot be sequenced. Upon encountering such a loop, the sorting algorithm is aborted, and all subsequent statements are executed in order.

The use of the sort algorithm places a severe restriction on the type of statements which can be used in the DERIVATIVE Region. No FORTRAN logic or branch statements can be used because they cannot be sorted properly. To allow the

versatility which accompanies these types of statements, once every integration interval, the program flow exits the DERIVATIVE Region and enters the DYNAMIC Region. The DYNAMIC Region provides an area in which FORTRAN logic can be used. Testing of variables, adjustment of coefficients and nonlinear operation all can be accomplished. It should be remembered, however, that since for a given value of the variable TIME, the DERIVATIVE Region is executed before the statements in the DYNAMIC Region a time lag equal to DELT exists between the actions taken in the DYNAMIC Region and the resulting effect in the DERIVATIVE Region.

A second system variable, DELS, dictates the frequency at which the computation interval is interrupted and program flow is diverted to the SAMPLE Region. The makeup of the SAMPLE Region is the same as that of the DYNAMIC Region. Its main function is to test variables to determine points for data collection and to order the program flow to allow simultaneous collection of data of several variables. DSLPLOT, which was installed at the Naval Postgraduate School, is accessed via the SAMPLE Region.

Finally, at the termination of the simulation either by a logic statement or when the variable TIME is set equal to the system variable FINTIM (a variable set by the user), the program makes one last pass through the SAMPLE Region and enters the TERMINAL Region. In the TERMINAL Region, logic can be used to modify program variables and constants and a new run initiated. If DSLPLOT has been utilized, the plotting

data set is closed out. If no action is taken to initiate a new run, the DSL job will be terminated. If a new run has been called for program flow returns to the INITIAL Region.

When the DSL job is completed, the DSL program is dynamically linked via the IBM 360 linkage editor to the program DSLPLOT.

C. DSLPLOT

The program DSLPLOT was produced as a forerunner to this thesis to allow the DSL output variables to be displayed via the CALCOMP XY Plotter. DSLPLOT accomplishes this by writing the variables to be plotted along with counters indicating the plot number, curve number, and individual point number on an external data set via the subroutine DRWG each sample interval (DELS). Each group of data points indicating a single DSL run is delineated by a set of zeroes produced by the subroutine ENDRW called in the TERMINAL Region.

PLOTHALL, the appending FORTRAN program used by DSLPLOT, accesses this data set upon a successful completion of the DSL job. A three-way sort is performed on the data set, and the points to be plotted are written in order on a second data set. The points are then retrieved one curve at a time from the second data set and passed to the plotting subroutine. This subroutine utilizes the IBM Plotting Package and produces the required CALCOMP plots.

Since it was intended that DSLPLOT was to be a permanent addition to the DSL program at the Naval Postgraduate School, it was written with as much flexibility as possible. Listed in Table A-1 are the various modes under which DSLPLOT can operate. The different modes allow the user to choose the size of the output plot and to either specify the scaling used on either axis or to allow the program to auto scale one or both axes.

D. USE OF THE PLOT PACKAGE

As previously discussed, points are collected for plotting via the subroutine DRWG. A call to this subroutine is initiated by the FORTRAN statement:

```
CALL DRWG(1,1, NUM,X,Y)
```

Where X and Y are the variables to be plotted. The X variable is plotted as the abscissa coordinate and Y the ordinate value. NUM is the integer counter denoting the data point number. The two integer constants appearing as the first two arguments represent the plot number and curve number for that plot number. These arguments may be provided as integer variables. This allows overlaying a set of curves from different runs on the same plot. It must be remembered that there is no convention in DSL, as there is in standard FORTRAN, for providing integer variables. Those variables such as NUM, or the variable NPLOT used in ENDRW must be declared integers by a DSL INTGER statement.

TABLE A-1

Mode No.	User Supply (in order listed)
0	Auto (no inputs required)
1	Xmin Dx Ymin Dy
2	Xmin Dx
3	Ymin Dy
4	Lenx Leny
5	Xmin Dx Ymin Dy Lenx Leny
6	Xmin Dx Lenx Leny
7	Ymin Dy Lenx Leny
8	Lenx
9	Xmin Dx Ymin Dy Lenx
10	Xmin Dx Lenx
11	Ymin Dy Lenx
12	Leny
13	Xmin Dx Leny
14	Xmin Dx Ymin Dy Leny
15	Ymin Dy Leny

Note: If mode 0 is specified a 9" by 9" graph with all values auto scales will be produced.

Upon completion of data collection for a given set of plots, the data set is closed out by the FORTRAN statement:

```
CALL ENDRW(NPLOT)
```

Where NPLOT is an integer variable which is initially set to the number of intended data sets. Each call to ENDRW decrements the variable by one. When NPLOT is less than or equal to zero, the DSL job is terminated. After a call to ENDRW, if NPLOT is still greater than zero, the program simply exits the subroutine, and the DSL program continues.

Each data set must be closed out by a call to ENDRW. If the plotting package has been used in the job, ENDRW must also be used as the means of terminating the DSL job. PLOTHALL, link edited to the DSL job, terminates upon encountering a value of NPLOT equal to one, therefore if the main DSL job is terminated by means other than ENDRW, an abnormal termination of the plot package will result.

Since it is standard practice to initiate a new set of plots each DSL run, the plot, curve and data point indicators refer to a given run. After a call to ENDRW, if a new run is initiated either as a parameter study or by a call to RERUN, CONTIN, etc., encountering the same DRWG statement again will generate a new plot. If curves are to be overlaid on one plot; at the end of a given run the data set must be held open by not calling ENDRW and the curve indicator, specified by a variable name, is updated. When the plot is complete,

ENDRW must then be called to close the data set. It is important to note that in any case, NUM, the data point counter must be initialized to 0 at the beginning of each run and incremented prior to its respective group of DRWG statements.

It is important also to remember that ENDRW merely closes out the current plot data set and decrements its calling argument NPLOT. ENDRW does not initiate a new DSL run. This must be done by the user with a call to RERUN, or CONTIN or by a parameter study or by any other method given in the DSL USER'S MANUAL. Figure A-1 is a sample outline of a DSL program, showing the use of the subroutines DRWG and ENDRW to produce plot output. In Figure A-1, two data sets are to be formed. Each data set will contain two plots. The first plot in each data set will contain two curves.

Another important point is that the subroutines DRWG and ENDRW are completely independent of the DSL statements PREPAR, GRAPH and PRTPLOT. The original IBM system output its data to the XY plotter via the statements GRAPH and PREPAR. At the Naval Postgraduate School, however, PREPAR, GRAPH and PRTPLOT (which at this installation must accompany GRAPH) are only used to produce the high-speed printer plots.

The plot package will handle up to ten plots per run and ten curves per plot. Any given curve can have up to 2000 points. If more than 2000 points are collected for a curve, the DSL run will be terminated, and the TERMINAL Region entered.

```
INTGER NUM,NPLOT
CONST NPLOT=2
CONTRL FINTIM=10.0,DELT=0.1,DELS=0.1
INITIAL
       -
       -
       -
       NUM=0
DERIVATIVE
       -
       -
SAMPLE
       NUM=NUM+1
       CALL DRWG(1,1,NUM,TIME,X)
       CALL DRWG(1,1,NUM,TIME,Y)
       CALL DRWG(2,1,NUM,X,Y)
       -
       -
TERMINAL
       CALL ENDRW(NPLOT)

           (parameter or constant changes
            using either FORTRAN assignment
            statements or a DSL END statement
            followed by the appropriate DSL
            non-executable statements)

       CALL RERUN (when FORTRAN assignments
                   have been used)
END
STOP
```

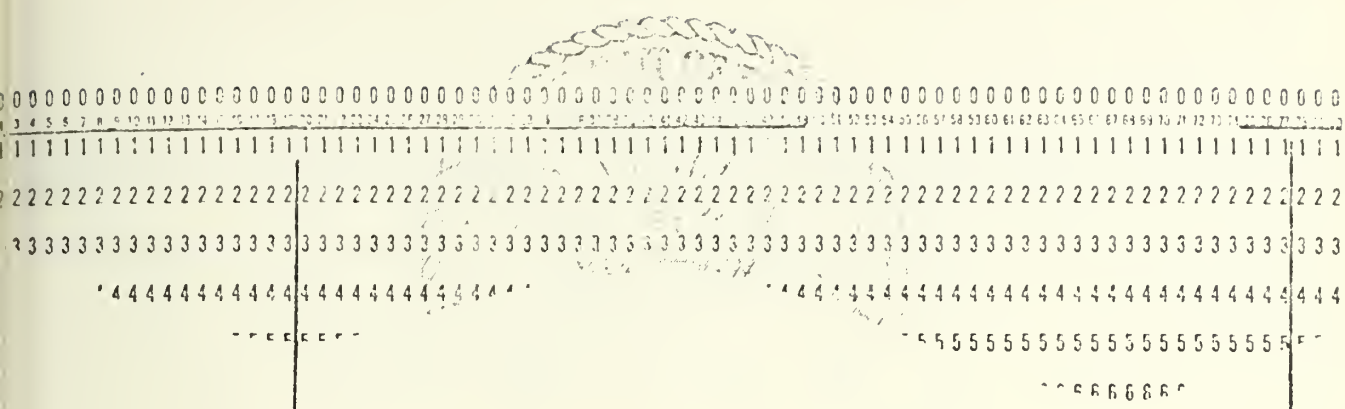
Figure A-1

Upon successful completion of the DSL job, the plot program PLOTHALL is executed. Each plot set up in the DSL job must be accompanied by two title cards. Columns 1 through 48 on both cards are reserved for titles. Columns 75 through 80 of the second title card constitutes the mode field (Figure A-2).

The mode numbers are listed in Table A-1. If a printed output is specified by supplying a printed output interval, then a listing of the points actually plotted will be provided sampled at the interval specified. The printed output interval can have a range from every point plotted to one of every 999 points plotted (assuming 999 or more points exist). If column 75 is left blank, no grid of scaled axes will be drawn; just a left-hand and bottom scale will be provided. If a 1 is placed in this column, a labeled grid at 3 inch intervals will be drawn. This grid is time consuming and should be omitted if not essential.

If a mode other than mode 0 is specified a third title card must be provided for the particular plot in question. The appropriate data must be entered via this title card as shown in Figure A-3.

Figure A-4 provides a listing of the JCL (job control language) cards necessary to implement a DSL job.

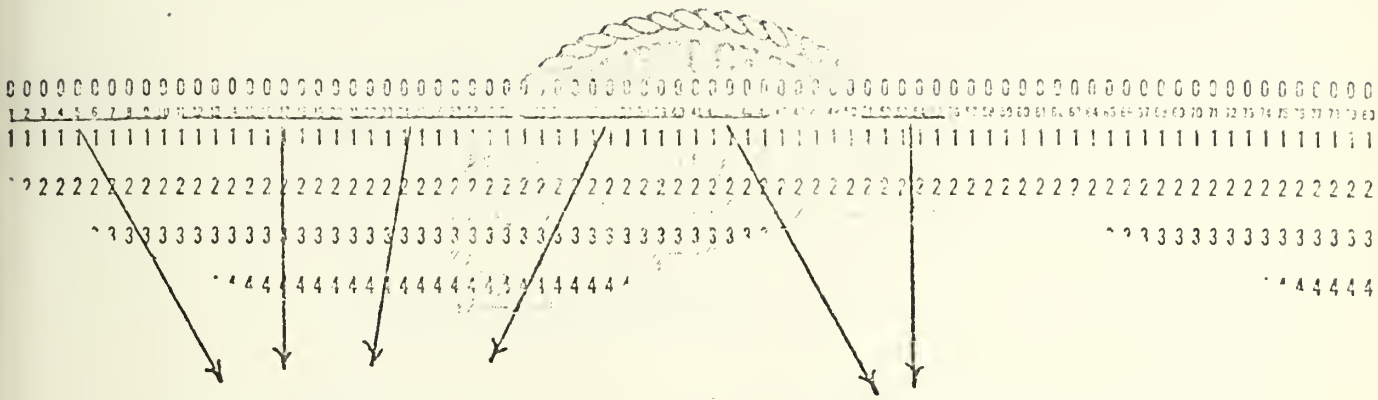


Columns 1 thru 48 on the first two title cards per plot is the title field. The user's name or box number should appear somewhere within this field on either card.

Mode Field:

Column 75: grid desired
Column 76-78: printed output interval
Column 79-80: mode number

Figure A-2



Scaling Fields

Graph Dimension Fields

Note: Columns 1 thru 40 are reserved for Xmin, Dx, Ymin and Dy (fields left justified as needed). Values may be input in F format or E format. If E format is used, data must be right justified in the appropriate field.

Note: Xmin is the value appearing on the extreme left end of the horizontal scale and Ymin is the value appearing on the bottom of the vertical scale. Dx and Dy are the respective scale increments. Lenx and Leny are the dimensions of the plot in inches in the X and Y directions, where:

$$0 < \text{Lenx} \leq 9.0$$

$$0 < \text{Leny} \leq 20.0$$

Obviously Dx and Dy must be compatible with Lenx and Leny. When no lengths are specified a 9.0" by 9.0" plot will be drawn.

Note: Columns 41 thru 45 and 51 thru 55 are reserved for Lenx and Leny (fields left justified as needed). Values must be input in F format.

Figure A-3

```
//    (standard green job card)
// EXEC DSL
//DSL.INPUT DD *

     (dsl deck goes here)

//PLOT.SYSIN DD *  (if plots are to be drawn)

     (2 or 3 title cards per plot requested)

/*
/*   (Note:  2 /*'s must be provided)
```

Figure A-4

**** APPENDIX B ****

THE FOLLOWING IS A LISTING OF :
1. DSL MISSILE SIMULATION
2. PLOTHALL
3. DSL PROGRAM FOR THE BODE PHASE AND MAGNITUDE PLOTS OF THE SELF ADAPTIVE FILTER
4. NOISE GENERATOR WITH ITS ACCOMPANYING FILTER SUBROUTINE

****

```
// EXEC DSL4,REGION=150K
//DSL.INPUT DD *

INTGER I,I1,I11
INTGER SYNCR1,NUM1
INTGER NUM2
INTGER NUM,IFIN2,NPLOT,IPLOT,NGR,FC
INTGER SYNC
INTGER SYNC1,SYNC2,SYNC3,SYNC4,SYNC5
INTEG RKSFX
CONTRL FINTIM=5.0,DELT=0.0001,DELS=0.0025
PRINT 0.002,C,B,BFO,HARM51,HARM52,HARM,SHIFT,THPR,FREQ
RANGE FREQ,TFREQ,B,BFO     ,BF01,HARM,HARM1,HARM4,BF
CONST IPLOT=1
CONST NPLOT=3
CONST SYNC5=0
*
INITIAL                                                                  CHA00020
*
PARAM ZETA=0.1
PARAM PI=3.141593
PARAM PUL1=0.0
PARAM PUL2=0.0
PARAM KTHDOT=1.75
PARAM P1=-55.0
PARAM P2=+55.0
PARAM CK1=3.12                                                           CHA00060
PARAM CK2=0.326                                                          CHA00070
PARAM KDELT1=-1.04
PARAM KDELT2=-0.78
PARAM FA=0.0                                                             CHA00100
PARAM FSQ=0.0                                                            CHA00110
PARAM CKF1=0.0                                                           CHA00120
PARAM CKF2=0.05                                                          CHA00130
STORAG ICB(3),CBT(2),CBB(4),ICI3(2),ICI4(2),CI2T(2),CI3T(2),...          CHA0
CI3B(3),CI4T(2),CI4B(3),CI1B(2),CI2B(3),ICH(2),CHT(1),CHB(3)   ,...      CHA00160
CHPT(3),CHPB(4),ICHP(4)     ,CGB(2),CGT(1),ICG(1),ICI2(2),...
   ICFIL(3),FILTP1(2),FILBT1(4),FILTP2(2),FILBT2(4),TFREQ1(6)
TABLE ICB(1-3)=3*0,CBT(1-2)=0.1,1.0,CBB(3-4)=1.0,0,ICH(1-2)=2*0,...      CHA00170
CHT(1)=1.0,ICI3(1-2)=2*0,ICI4(1-2)=2*0,ICHP(1-4)=4*0,ICI2(1-2)=2*0,...
ICFIL(1-3)=3*0.0,FILTP1(1)=1.0,FILTP1(2)=0.0,FILBT1(1)=1.0,...
```

```
FILTP2(1)=1.0,FILTP2(2)=0.0,FILBT2(1)=1.0
      CBB1=470.0                                                    CHA00190
      CBB(1)=1.0/CBB1**2                                            CHA00200
      CBB(2)=(2.0*0.3)/470.0                                        CHA00210
      CHB1=440.0                                                    CHA00220
      CHB(1)=1.0/CHB1**2                                            CHA00230
      CHB(2)=(2.0*0.6)/440.0                                        CHA00240
      CHB(3)=1.0                                                    CHA00250
      ICG(1)=0.0
      ICI1=0.0
      A11=0.151
      A1=0.01
      B1=1.0/2000.0
      A22=(1.0/495.0)**2
      A12=2.0*0.05/495.0
      B22=(1.0/460.0)**2
      B12=2.0*0.3/460.0
      A23=(1.0/1000.0)**2
      A13=2.0*0.05/1000.0
      B23=(1.0/1000.0)**2
      B13=2.0*0.7/1000.0
      A24=(1.0/755.0)**2
      A14=2.0*0.05/755.0
      B24=(1.0/680.0)**2
      B14=2.0*0.4/680.0
      CGT(1)=1000.0
      CGB(1)=1.0
      CGB(2)=1000.0
      CHPT(1)=A11*A1
      CHPT(2)=A11
      CHPT(3)=0.0
      CHPB(1)=B1*CHB(1)
      CHPB(2)=B1*CHB(2)+CHB(1)
      CHPB(3)=B1+CHB(2)
      CHPB(4)=1.0
      CI2T(1)=A12-A22*B12/B22
      CI2T(2)=1.0-A22/B22
      CI2B(1)=B22
      CI2B(2)=B12
      CI2B(3)=1.0
      CI3T(1)=A13-A23*B13/B23
      CI3T(2)=1.0-A23/B23
      CI3B(1)=B23
      CI3B(2)=B13
      CI3B(3)=1.0
      CI4T(1)=A14-A24*B14/B24
      CI4T(2)=1.0-A24/B24
      CI4B(1)=B24
```

```
      CI4B(2)=B14
      CI4B(3)=1.0
*
***     INITIALIZING FILTER COEFFICIENTS
*
      FREQ=26.3
      FILGA=3000.0
      FREQ1=2.0*PI*FREQ
      FREQ2=2.0*FREQ1
      FREQ3=3.0*FREQ1
      FILTP1(3)=FREQ2**2
      FILBT1(2)=2.0*ZETA*FREQ2+FILGA
      FILBT1(3)=FREQ2**2+2.0*ZETA*FREQ2*FILGA
      FILBT1(4)=FREQ2**2*FILGA
      FILTP2(3)=FREQ3**2
      FILBT2(2)=2.0*ZETA*FREQ3+FILGA
      FILBT2(3)=FREQ3**2+2.0*ZETA*FREQ3*FILGA
      FILBT2(4)=FREQ3**2*FILGA
*
***   HARMONIC AMPLITUDE
*
      HMAG=0.1
      HMAG2=1.5
   69 FREQT=1.0/(2.0*PI*10.0)
*
***     INITIALIZING ALL VARIABLES TO ZERO
*
      A=0.0
      AI=0.0
      A2=0.0
      B=0.0
      C=0.0
      D=0.0
      F=0.0
      G4=0.0
      H=0.0
      J=0.0
      THC=0.0
      THPR=0.0
      DELC=0.0
      DELNS=0.0
      NUM=0
      NUM1=0
      NUM2=0
      SYNC=0
      IFIN2=0
      SYNC1=0
      SYNC2=0
```

```
      SYNC3=1
      T2=0.0
      T6=0.0
      TFREQ2=0.0
*
      CALL PRINT
*
**    SET FLIGHT CONDITIONS
*
   15 FC=15
      MACH=2.34
      GAIN1=2526.127
      PRESS=573.279
      VELA=986.405
      IR=4.9
      CRO=0.185
      GO TO 17
   16 FC=24
      MACH=2.06
      GAIN1=7750.100
      PRESS=2116.22
      VELA=1116.89
      IR=5.3
      CRO=0.20
*
**    CALCULATING VALUES FOR G AND F
*
   17 LAMDA=PRESS/2116.22
      VEL=MACH*VELA
      POLE1=1481.0*LAMDA**0.994*1.125*(MACH**2)*(57.3/IR)*(1.125/VEL)*CRO
      POLE=1.0/POLE1
      GAIN=GAIN1/POLE1
      WRITE(6,500)FC,POLE1,GAIN1
  500 FORMAT('0','FOR FC#',I4,1X,'F=',F10.6,1X,'G=',F12.5)
*
DERIVATIVE
*
**    A BLOCK
*
      AI=THPR-THC
      A2=LIMIT(P1,P2,AI)/CK2
      A=J+A2
*
**    B BLOCK
*
      BI=TRNFR(1,3,ICB,CBT,CBB,A)
      BFO=KDELT2*BI
      BFIL1=REALPL(0.0,FREQT,BFO)
```

```
      BFIL2=REALPL(0.0,FREQT,BFIL1)
       BFDC=REALPL(0.0,FREQT,BFIL2)
      BFO1=BFO-BFDC
*
*     HARMONIC GENERATOR
*
      HARM1=(HMAG*BFO1)**2
      HARM2=REALPL(0.0,FREQT,HARM1)
      HARM3=REALPL(0.0,FREQT,HARM2)
      HARM4=REALPL(0.0,FREQT,HARM3)
      HARM5=HARM1-HARM4
PROCED HARM,HARM51,HARM52,SHIFT=ROLL(HMAG2,HARM5,SYNC5,TIME,FREQ,NPLOT)
      HARM51=HMAG2*HARM5
      SHIFT=(1.0/FREQ)*(1.0+SINE(0.0,62.8,0.0))
      HARM52=DELAY(300,SHIFT,HARM51)
      HARM=HARM52
      IF((TIME.LT.1.0).OR.(SYNC5.EQ.0)) HARM=0
ENDPRO
      BF=BFO+HARM
*
*     INSERTING FILTER
*
PROCED B=FILTER(BF,FILOUT)
      B=BF
      IF(SYNC.EQ.1) B=FILOUT
ENDPRO
*     BISTABLE ELEMENT ( C BLOCK )
*
PROCED C=BISTBL(B,FA,FSQ)
      C1=B+FA+FSQ
   1  IF(C1.GT.0.0) GO TO 3                                          HDERV  9
      C=-200.0                                                       HDERV 10
      GO TO 4                                                        HDERV 11
   3  C=200.0                                                        HDERV 12
   4  CONTINUE
ENDPRO
*
*     D BLOCK
*
      D=DELAY(40,0.00455,C)
*
*     F BLOCK
*
      F1=CKF1*CKF2*F+DELNS+D                                         HDERV 16
      F=INTGRL(0,F1)                                                 HDERV 17
*
*     FILTER BLOCK
*
```

```
      E1DOT1=E21
      E11=INTGRL(0.0,E1DOT1)
      E2DOT1=E31
      E21=INTGRL(0.0,E2DOT1)
      E3DOT1=BF-FILBT1(2)*E31-FILBT1(3)*E21-FILBT1(4)*E11
      E31=INTGRL(0.0,E3DOT1)
      FIL1=E31+FILTP1(3)*E11
      E1DOT2=E22
      E12=INTGRL(0.0,E1DOT2)
      E2DOT2=E32
      E22=INTGRL(0.0,E2DOT2)
      E3DOT2=FIL1-FILBT2(2)*E32-FILBT2(3)*E22-FILBT2(4)*E12
      E32=INTGRL(0.0,E3DOT2)
      FIL2=E32+FILTP2(3)*E12
      FILOUT=FIL2*FILGA**2
*
**    G BLOCK
*
      G2=F-DELC
      G1=REALPL(0.0,POLE,G2)
      G=GAIN*G1
*
**    H BLOCK
*
      H=TRNFR(0,2,ICH,CHT,CHB,G)
*
**    HP BLOCK
*
      HP=TRNFR(2,3,ICHP,CHPT,CHPB,G)
*
**    I BLOCK
*
      I2PR=TRNFR(1,2,ICI2,CI2T,CI2B,HP)
      I2=(A22/B22)*HP+I2PR
      I3PR=TRNFR(1,2,ICI3,CI3T,CI3B,I2)
      I3=(A23/B23)*I2+I3PR
      I4PR1=TRNFR(1,2,ICI4,CI4T,CI4B,I3)
      I41=(A24/B24)*I3+I4PR1
      I4PR2=TRNFR(1,2,ICI4,CI4T,CI4B,I41)
      I4=(A24/B24)*I41+I4PR2
*
**    J BLOCK
*
      J1=KTHDOT*H+I4
      J=0.331*J1                                                    HDERV 27
      THPR=INTGRL(0,G)                                              HDERV 29
*
DYNAMIC
```

```
*
**      WHITE NOISE SOURCE
*
        IF(NPLOT.EQ.1) SYNC5=1
        IF(NPLOT.LE.2) SYNC=1
        DELNS=0.0
        GO TO 35
        IF(NPLOT.EQ.1) SYNC=1
        IF(TIME.LT.1.0) GO TO 35
        CALL NOISE(DELNS)
        DELNS=0.1*DELNS
*
**      MEASURING FREQUENCY
*
   35   IF(TIME.LT.0.4) RETURN
        IF(C.GT.100.0) SYNC1=1
        IF((SYNC1.EQ.1).AND.(SYNC2.EQ.1).AND.(T2.NE.0.0)) T6=TIME
        IF((SYNC1.EQ.1).AND.(SYNC2.EQ.1).AND.(T2.EQ.0.0)) T2=TIME
        IF(SYNC1.EQ.1) SYNC2=0
        IF(C.LT.-100.0) SYNC2=1
        IF(SYNC2.EQ.1) SYNC1=0
        IF((T2.NE.0.0).AND.(T6.NE.0.0)) GO TO 60
        RETURN
   32   FREQ1=2.0*PI*FREQ
        FREQ2=2.0*FREQ1
        FREQ3=3.0*FREQ1
        FILTP1(3)=FREQ2**2
        FILBT1(2)=2.0*ZETA*FREQ2+FILGA
        FILBT1(3)=FREQ2**2+2.0*ZETA*FREQ2*FILGA
        FILBT1(4)=FREQ2**2*FILGA
        FILTP2(3)=FREQ3**2
        FILBT2(2)=2.0*ZETA*FREQ3+FILGA
        FILBT2(3)=FREQ3**2+2.0*ZETA*FREQ3*FILGA
        FILBT2(4)=FREQ3**2*FILGA
        RETURN
*
**      AVERAGING FREQUENCY
*
   60   TFREQ=1.0/(T6-T2)
        IF(SYNC3.EQ.7) GO TO 61
        SYNC4=7-SYNC3
        TFREQ1(SYNC4)=TFREQ
        SYNC3=SYNC3+1
        GO TO 62
   61   CONTINUE
        DO 63 I=1,5
        I1=7-I
        I11=6-I
```

```
   63 TFREQ1(I1)=TFREQ1(I11)
      TFREQ1(1)=1.0/(T6-T2)
      DO 64 I=1,6
   64 TFREQ2=TFREQ2+TFREQ1(I)
      FREQ=TFREQ2/6.0
      TFREQ2=0.0
   62 T2=0.0
      T6=0.0
      GO TO 32
      CONTINUE
*
SAMPLE
      NUM2=NUM2+1
      CALL DRWG(1,1,NUM2,TIME,THPR)
TERMINAL
      CALL ENDRW(NPLOT)
   49 CALL RERUN
END
STOP


FORTRAN
      SUBROUTINE NOISE(TRASH)
      READ(3,100,END=1) TRASH,NGR
  100 FORMAT(1X,F14.8,I6)
      RETURN
    1 WRITE(6,101) NGR
  101 FORMAT('0','END OF NOISE DATA.',I6,1X,'NOISE DATA POINTS USED.REPE
     1ATING SAME NOISE RECORD.')
      REWIND 3
      READ(3,100) TRASH,NGR
      RETURN
      END


FORTRAN
      SUBROUTINE PRTOUT(ICOUNT,NUM)
      IF(NUM.LT.ICOUNT) GO TO 1
      CALL PRINT
      NUM=1
      RETURN
    1 NUM=NUM+1
      RETURN
      END


//G.FT04F001 DD SPACE=(CYL,(5,1)),DCB=BLKSIZE=1998
//G.FT03F001 DD DUMMY
```

```
//PLOT.FT12F001 DD SPACE=(CYL,(3,1))
//PLOT.SYSIN DD *
NOTE: TITLES APPEAR AFTER PLOT.SYSIN CARD
```

PLOTHALL IS RESIDENT ON DISK AND IS DYNAMICALLY LINK EDITED TO A MAIN DSL PROGRAM AT TIME OF EXECUTION TO PRODUCE THE DSL PLOT PACKAGE. TO ALLOW THE EDITING TO BE ACCOMPLISHED NORMALLY EVERYTIME REGARDLESS OF WHETHER OR NOT A PLOT IS DESIRED, PLOTHALL INITIALLY MODIFYS THE PLOT DATA SET BY WRITING A FLAG WORD AT THE END OF THE DATA SET. PLOTHALL THEN READS THE FIRST WORD IN THIS DATA SET, IF THIS IS THE FLAG WORD THE PLOT PROGRAM IS TERMINATED.

****

```
// EXEC FORTCL
//FORT.SYSIN DD *

      INTEGER COUNT1,EOF
      INTEGER*2 NCR(10),NUM(10,10)
      DIMENSION PLOTX(2000),PLOTY(2000),PLOTAX(2000),PLOTAY(2000),PLOTBX
     1(2000),PLOTBY(2000),PLOTCX(2000),PLOTCY(2000)
      COMMON PLOTCX,PLOTCY
      EQUIVALENCE(PLOTX(1),PLOTAX(1)),(PLOTY(1),PLOTAY(1))
      REAL*8 ITITLE(12),BLANK/'          '/
      REAL LABEL(15)
      DATA                LABEL    /'1   ','2   ','3   ','4   ','5   ','6
     1   ','7   ','8   ','9   ','10  '/
      DATA EOF/9/,IPLOT1/0/
      WRITE(4,701) EOF
  701 FORMAT(I1)
      REWIND 4
      NZERO=0                                                        DSL ALT
      ICOUNT=0
      IC1=1
   43 CCUNT1=0
      DO 645 I=1,10
  645 NCR(I)=0
  600 READ(4,124,END=992)IEOF,IPLOT,NGR,NUMX
  124 FORMAT(I1,I2,I2,I4)
      IF(IEOF.EQ.EOF) GO TO 992
      IF(NUMX.EQ.0) COUNT1=COUNT1+1
      IF((NUMX.EQ.0).AND.(COUNT1.GT.NZERO)) GO TO 611
      IF((IPLOT.GT.10).OR.(NGR.GT.10)) GO TO 24
      IF((NGR.GT.NCR(IPLOT)).AND.(COUNT1.EQ.NZERO)) NCR(IPLOT)=NGR
      GO TO 600
  611 NPLOT=IPLOT
      IPLOT1=0
      DO 612 I=1,10
      IF(NCR(I).EQ.0) GO TO 613
  612 IPLOT1=I
  613 IF(IPLOT1.EQ.0) GO TO 992
      REWIND 4
      CCUNT1=0
      J=1
      NCR1=1
  614 IF(COUNT1.EQ.NZERO) GO TO 636
      READ(4,101) NUMX
  101 FORMAT(5X,I4)
      IF(NUMX.EQ.0) COUNT1=COUNT1+1
      GO TO 614
  636 NCR2=0
```

```
      NCR3=0
      IPLOTA=0
      IPLOTB=0
      IPLOTC=0
      IF(NCR1.LE.NCR(J)) IPLOTA=J
      IF(IPLOTA.EQ.0) GO TO 635
      IF(NCR1.LT.NCR(J)) NCR2=NCR1+1
      IF((NCR2.LT.NCR(J)).AND.(NCR2.NE.0)) NCR3=NCR2+1
      IF(NCR2.NE.0) IPLOTB=J
      IF(NCR3.NE.0) IPLOTC=J
      IF(IPLOTB.EQ.0) GO TO 618
      IF((IPLOTB.NE.0).AND.(IPLOTC.EQ.0)) GO TO 619
  620 I1=1
      I2=1
      I3=1
  633 READ(4,100) IPLOT,NGR,NUMPT,DSLX,DSLY
  100 FORMAT(1X,I2,I2,I4,2E14.8)
      IF((IPLOT.NE.IPLOTA).OR.(NGR.NE.NCR1)) GO TO 630
      NUM(IPLOT,NGR)=NUMPT
      PLOTAX(I1)=DSLX
      PLOTAY(I1)=DSLY
      I1=I1+1
      GO TO 632
  630 IF((IPLOT.NE.IPLOTB).OR.(NGR.NE.NCR2)) GO TO 631
      NUM(IPLOT,NGR)=NUMPT
      PLOTBX(I2)=DSLX
      PLOTBY(I2)=DSLY
      I2=I2+1
      GO TO 632
  631 IF((IPLOT.NE.IPLOTC).OR.(NGR.NE.NCR3)) GO TO 632
      NUM(IPLOT,NGR)=NUMPT
      PLOTCX(I3)=DSLX
      PLOTCY(I3)=DSLY
      I3=I3+1
  632 IF(NUMPT.NE.0) GO TO 633
      REWIND 4
      CCUNT1=0
      NOPT=NUM(IPLOTA,NCR1)
      DO 639 I=1,NOPT
  639 WRITE(12,700) PLOTAX(I),PLOTAY(I)
  700 FORMAT(2E14.8)
      IF(IPLOTB.EQ.0) GO TO 637
      NOPT=NUM(IPLOTB,NCR2)
      DO 641 I=1,NOPT
  641 WRITE(12,700) PLOTBX(I),PLOTBY(I)
      IF(IPLOTC.EQ.0) GO TO 637
      NOPT=NUM(IPLOTC,NCR3)
      DO 642 I=1,NOPT
```

```
642 WRITE(12,700) PLOTCX(I),PLOTCY(I)
637 J=0
    IF(IPLOTC.NE.0) J=IPLOTC
    IF(J.EQ.0) GO TO 634
    NCR1=NCR3+1
    IF(NCR1.GT.NCR(J)) GO TO 635
    GO TO 614
618 IPLOTB=J+1
    IF(IPLOTB.LE.IPLOT1) NCR2=1
    IF(NCR2.EQ.0) IPLOTB=0
    IF(IPLOTB.EQ.0) GO TO 620
    IF(NCR2.LT.NCR(IPLOTB)) NCR3=NCR2+1
    IF(NCR3.NE.0) IPLOTC=IPLOTB
    IF(IPLOTC.EQ.0) GO TO 622
    GO TO 620
619 IPLOTC=J+1
624 IF(IPLOTC.LE.IPLOT1) NCR3=1
    IF(NCR3.EQ.0) IPLOTC=0
    GO TO 620
622 IPLOTC=IPLOTB+1
    GO TO 624
635 J=J+1
    IF(J.GT.IPLOT1) GO TO 634
    NCR1=1
    GO TO 614
634 REWIND 12
    DO 39 J=1,IPLOT1
    NUM3=0
    READ(5,1,END=991)(ITITLE(I),I=1,6)
    READ(5,2,END=991) (ITITLE(I),I=7,12),MODE3,MODE1,MODE
  1 FORMAT(6A8)                                                    HALLDC20
2   FORMAT(6A8,26X,I1,I3,I2)
    IF(MODE.EQ.0) GO TO 31
    GO TO 29
 31 ICOUNT=ICOUNT+1
    NGR1=NCR(J)
    DO 6 INGR=1,NGR1
    NUM1=NUM(J,INGR)
    MODE2=MODE1
    IF(MODE1.GE.NUM1) MODE2=NUM1/10
    IF(MODE2.EQ.0) MODE2=1
    DO 643 K=1,NUM1
643 READ(12,700,END=994) PLOTX(K),PLOTY(K)
996 NUM3=NUM3+NUM1
    NCUR=1
    IF(NGR1.EQ.1) GO TO 8
    IF((INGR.NE.NGR1).AND.(INGR.NE.1)) GO TO 25
    IF(INGR.EQ.NGR1) NCUR=3
```

```
      GO TO 26
   25 NCUR=2
   26 IF(MODE1.EQ.0) GO TO 800
      WRITE(6,123) MODE2,INGR,J,IC1
  123 FORMAT('0','POINTS PLOTTED(1 OF ',I3,') FOR CURVE#',I2,2X,'OF PLOT
     1#',I3,2X,'FOR DSL RUN#',I2/'+',66('_')/)
      DO 7 I=1,NUM1,MODE2
    7 WRITE(6,102) PLOTX(I),PLOTY(I)
  102 FORMAT(' ',E14.8,5X,E14.8)
  800 CALL DRWG(NUM1,PLOTX,PLOTY,NCUR,LABEL(INGR),ITITLE,XMIN,DX,YMIN,
     1DY,MODE,YLEN,J,IC1,XLEN,MODE3)
    6 CONTINUE
      GO TO 9
    8 NGRX=1
      IF(MODE1.EQ.0) GO TO 801
      WRITE(6,123) MODE2,NGRX,J,IC1
      DO 10 I=1,NUM1,MODE2
   10 WRITE(6,102)PLOTX(I),PLOTY(I)
  801 NCUR=0
      CALL DRWG(NUM1,PLOTX,PLOTY,NCUR,LABEL(NGR1),ITITLE,XMIN,DX,YMIN,
     1DY,MODE,YLEN,J,IC1,XLEN,MODE3)
    9 WRITE(6,115)J,IC1
  115 FORMAT('0',5X,'*** PLOT STATISTICS FOR PLOT#',I3,2X,'OF DSL RUN#',
     1I2,1X,'***')
      WRITE(6,116) NPLOT,MODE,NGR1,NUM3,(NUM(J,I),I=1,NGR1)
  116 FORMAT('0',5X,'NPLOT NO.=',I4/' ',5X,'MODE NO.=',I4/' ',5X,'NUMBER DSL 13T
     1 OF CURVES THIS PLOT =',I4/' ',5X,'TOTAL NUMBER OF POINTS THIS PLO DSL 13T
     2T =',I6/' ',5X,'NUMBER OF POINTS PER CURVE =',10(I4,3X))           DSL 13T
      IF(J.EQ.IPLOT1)WRITE(6,121)J
  121 FORMAT(' ',5X,'NUMBER PLOTS GENERATED BY THIS DSL RUN=',I4)
      WRITE(6,117)
  117 FORMAT('0',5X,'*********')
   39 CONTINUE
      REWIND 12                                                          DSL 13T
      IF(NPLOT.EQ.1) GO TO 42
      IC1=IC1+1
      NZERO=NZERO+1                                                      DSL 13T
      GO TO 43
   24 WRITE(6,103)
  103 FORMAT('0','*** ERROR:TOO MANY CURVES PER PLOT OR TOO MANY PLOTS P
     1ER RUN.'/' ',5X,'LIMIT IS 10 PLOTS PER RUN AND 10 CURVES PER PLOT.
     2 ***'/' ','*** JOB TERMINATED ***')
      STOP
   29 GO TO(30,37,32,51,52,53,54,55,56,57,58,59,60,33,34),MODE
      WRITE(6,111)MODE
  111 FORMAT('0','MODE=',I1,1X,'IS AN INCORRECT MODE NO.'/' AUTO SCALING
     1 ASSUMED.')
      MODE=0
```

```
      GO TO 31
 30   READ(5,106,END=993) XMIN,DX,YMIN,DY
106   FORMAT(4G10.3)
      GO TO 31
37    READ(5,105,END=993) XMIN,DX                                          DSL 13T
      GO TO 31                                                             DSL 13T
 32   READ(5,105,END=993)YMIN,DY
105   FORMAT(2G10.3)
      GO TO 31
 51   READ(5,130,END=993) XLEN,YLEN
130   FORMAT(40X,2(F5.2,5X))
      GO TO 35
 52   READ(5,125,END=993)XMIN,DX,YMIN,DY,XLEN,YLEN
125   FORMAT(4(G10.3),2(F5.2,5X))
      GO TO 35
 53   READ(5,126,END=993)XMIN,DX,XLEN,YLEN
126   FORMAT(2(G10.3),20X,2(F5.2,5X))
      GO TO 35
 54   READ(5,126,END=993)YMIN,DY,XLEN,YLEN
      GO TO 35
 55   READ(5,127,END=993)XLEN
127   FORMAT(40X,F5.2)
      GO TO 62
 56   READ(5,112,END=993)XMIN,DX,YMIN,DY,XLEN
      GO TO 62
 57   READ(5,113,END=993)XMIN,DX,XLEN
      GO TO 62
 58   READ(5,113,END=993)YMIN,DY,XLEN
      GO TO 62
 59   READ(5,127,END=993)YLEN
      GO TO 35
 60   READ(5,113,END=993)XMIN,DX,YLEN
      GO TO 35
33    READ(5,112,END=993) XMIN,DX,YMIN,DY,YLEN
112   FORMAT(4G10.3,F5.1)
      GO TO 35
 34   READ(5,113,END=993) YMIN,DY,YLEN
113   FORMAT(2G10.3,20X,F5.2)
 35   IF(YLEN.GT.20.0) YLEN=20.0
      IF(YLEN.LE.0.0) GO TO 36
      IF((MODE.GT.11).OR.(MODE.LT.4)) GO TO 31
62    IF(XLEN.GT.9.0) XLEN=9.0
      IF(XLEN.LE.0.0) GO TO 61
      GO TO 31
36    WRITE(6,129)YLEN
129   FORMAT('0','LENGTH EITHER INCORRECTLY ENTERED OR OMITTED'/' ','Y-L
     1ENGTH=',F5.2/' ','AUTO SCALING ASSUMED')
      MODE=0
```

```
      GO TO 31
61    WRITE(6,128) XLEN
128   FORMAT('0','LENGTH EITHER INCORRECTLY ENTERED OR OMITTED'/' ','X-L
     1ENGTH=',F5.2/' ','AUTO SCALING ASSUMED')
      MODE=0
      GO TO 31
991   WRITE(6,107)
107   FORMAT('0','ERROR:ONE OR MORE TITLE CARDS ARE MISSING'/' *** JOB T
     1ERMINATED ***')
      STOP
992   IF((ICOUNT.EQ.0).AND.(IPLOT1.EQ.0)) GO TO 41
      WRITE(6,108)
108   FORMAT('0','ERROR:SUBROUTINE ENDRW WAS NOT USED TO TERMINATE THE M
     1AIN DSL PROGRAM'/' *** JOB TERMINATED ***')
      STOP
41    WRITE(6,120)
120   FORMAT('0','*** NO DATA SET FORMED REQUESTING CALCOMP PLOTS ***'/1
     16X,'*** END OF DSLPLOT ***')
      STOP
993   WRITE(6,110)
110   FORMAT('0','MANUAL SCALING REQUESTED BUT ONE OR MORE INPUTS ARE MI
     1SSING.'/' AUTO SCALING ASSUMED.')
      MODE=0
      GO TO 31
42    WRITE(6,122) IC1,ICOUNT
122   FORMAT('0',5X,'*** DSLPLOT STATISTICS ***'/'0',5X,'TOTAL NUMBER OF
     1 DSL RUNS=',I4/' ',5X,'TOTAL NUMBER OF PLOTS GENERATED=',I5/'0',5X
     2,'********')
      STOP
994   WRITE(6,995) NUM1,J,INGR,NGR1,IPLOT1
995   FORMAT(' ','ERROR:NUM1=',I4,'J=',I3,'INGR=',I3,'NGR1=',I3,'IPLOT1=
     1',I3)
      GO TO 996
      END


      SUBROUTINE DRWG(NUM2,PLTDX,PLTDY,NCUR,LABEL,ITITLE,XMIN,DX,YMIN,
     1DY,MODE,YLEN,ICOUNT,IRUN,XLEN,MODE1)
      DIMENSION PLOTX(2000),PLOTY(2000),TITLE1(7),TITLE2(7),TITLE3(2),
     1PLTDX(2000),PLTDY(2000),TITLE4(2),TITLE5(2)
      COMMON PLOTX,PLOTY
      REAL*8 ITITLE(12)
      REAL TITLE/'    '/
      REAL LABEL
      DATA TITLE1/'XSCA','LE= ','    ','    ',' UNI','TS/I','NCH '/
      DATA TITLE2/'YSCA','LE= ','    ','    ',' UNI','TS/I','NCH '/
      DATA TITLE3/'PLOT',' NO.'/
      DATA TITLE4/'****','****'/
```

```
      DATA TITLE5/'RUN ','NO. '/
      DATA TITLE6/'E   '/
      IF(NCUR.EQ.0) NCUR=4
      IFLAG=0
      IF((MODE.LT.4).OR.((MODE.GT.7).AND.(MODE.LT.12))) GO TO 26
   29 IF((MODE.LT.4).OR.(MODE.GT.11)) GO TO 28
   25 YMAX1=YLEN+0.1
      XMAX1=XLEN+0.1
      RUN=IRUN
      COUNT=ICOUNT
      NUM3=NUM2-1
      K=1
      DO 18 I=1,NUM3
      J=I+1
      IF((PLTDX(I).EQ.PLTDX(J)).AND.(PLTDY(I).EQ.PLTDY(J)))GO TO 19
      PLOTX(K)=PLTDX(I)
      PLOTY(K)=PLTDY(I)
      K=K+1
      IF(IFLAG.EQ.1) GO TO 20
   18 CONTINUE
      PLOTX(K)=PLTDX(NUM2)
      PLOTY(K)=PLTDY(NUM2)
      NUM1=K
      GO TO(11,11,17,17,11,11,17,17,11,11,17,17,11,11,17),MODE
   17 CONTINUE
      GO TO (35,11,11,35),NCUR
   35 CALL SCALE(PLOTX,NUM1,1,XLEN,0.0,XMIN,DX)
      GO TO(2,27,2,27,2,27,2,27,2,27,2,27,2,27,2),MODE
   27 CONTINUE
      GO TO (1,2,2,1),NCUR
    1 CALL SCALE(PLOTY,NUM1,1,YLEN,0.0,YMIN,DY)
    9 CALL PLOTS
      CALL SYMBOL(0.0,0.0,0.14,TITLE2,0.0,28)
      CALL SYMBOL(0.0,0.25,0.14,TITLE1,0.0,28)
      IF(DY.GE.1.0E06) GO TO 21
      IF(DY.LT.0.01) GO TO 32
   23 IF(DX.GE.1.0E06) GO TO 22
      CALL NUMBER(0.84,0.0,0.14,DY,0.0,2)
   24 XP1=XLEN-2.0
      CALL NUMBER(0.84,0.25,0.14,DX,0.0,2)
      IF(DX.LT.0.01) GO TO 33
      IF(XP1.LT.4.0) XP1=4.0
      XP2=XP1+0.96
      XP3=XP1+0.84
      CALL SYMBOL(XP1,0.0,0.14,TITLE3,0.0,8)
      CALL NUMBER(XP2 ,0.0,0.14, COUNT,0.0,-1)
      CALL SYMBOL(XP1,0.25,0.14,TITLE5,0.0,8)
      CALL NUMBER(XP3 ,0.25,0.14,RUN,0.0,-1)
```

```
      CALL PLOT(0.0,0.90,-3)
14    AX=0.0
      LENY=YLEN/3.0
      IF(MODE1.EQ.0) LENY=1
      IF(LENY.EQ.0) GO TO 30
      DO 16 I=1,LENY
      CALL AXIS(0.0,AX,TITLE,-4,XLEN,0.0,XMIN,DX)
16    AX=AX+3.0
      IF(MODE1.EQ.0) GO TO 34
 30   CALL AXIS(0.0,YLEN,TITLE,-4,XLEN,0.0,XMIN,DX)
 34   AX=0.0
      LENX=XLEN/3.0
      IF(MODE1.EQ.0) LENX=1
      IF(LENX.EQ.0) GO TO 31
      DO 15 I=1,LENX
      CALL AXIS(AX,0.0,TITLE,-4,YLEN,90.0,YMIN,DY)
15    AX=AX+3.0
      IF(MODE1.EQ.0) GO TO 5
 31   CALL AXIS(XLEN,0.0,TITLE,-4,YLEN,90.0,YMIN,DY)
 5    CALL LINE(PLOTX,PLOTY,NUM1,1,1)
      CALL WHERE(PX,PY)
      CALL SYMBOL(PX,PY,0.14,LABEL,0.0,4)
      IF((NCUR.EQ.3).OR.(NCUR.EQ.4)) GO TO 3
      RETURN
 3    SYM1=YLEN+1.0
      SYM2=YLEN+0.5
      CALL SYMBOL(0.0,SYM1,0.21,ITITLE(1),0.0,48)
      CALL SYMBOL(0.0,SYM2,0.21,ITITLE(7),0.0,48)
      CALL PLOT(0.0,YLEN+3.5,-3)
      CALL PLOTE
      WRITE(6,117)
117   FORMAT('0',5X,'********')
      WRITE(6,6)(ITITLE(I),I=1,6)
      WRITE(6,6)(ITITLE(I),I=7,12)
 6    FORMAT(5X,6A8)
      WRITE(6,7)
 7    FORMAT('0',5X,'GRAPH WITH THIS TITLE HAS BEEN DRAWN')
      RETURN
 2    DO 4 I=1,NUM1
      PLOTY(I)=(PLOTY(I)-YMIN)/DY
      IF(PLOTY(I).LT.0.0) PLOTY(I)=-0.1
 4    IF(PLOTY(I).GT.YLEN) PLOTY(I)=YMAX1
      IF(MODE.NE.0) GO TO 10
      GO TO 5
 10   GO TO (9,5,5,9),NCUR
11    DO 12 I=1,NUM1
      PLOTX(I)=(PLOTX(I)-XMIN)/DX
      IF(PLOTX(I).LT.0.0) PLOTX(I)=-0.1
```

```
   12  IF(PLOTX(I).GT.XLEN) PLOTX(I)=XMAX1
       GO TO(2,27,2,27,2,27,2,27,2,27,2,27,2,27,2),MODE
       GO TO 27
    19 IF(IFLAG.EQ.0) ICNT1=I
       IFLAG=1
       GO TO 18
    20 WRITE(6,118) ICNT1,I
   118 FORMAT(' ','*** WARNING ***',5X,'POINTS',I4,1X,'THRU',I4,1X,'ARE I
      1DENTICAL.ONLY THE LAST POINT IS ACTUALLY PLOTTED')
       IFLAG=0
       GO TO 18
    21 CALL SYMBOL(0.96,0.0,0.14,TITLE4,0.0,8)
       GO TO 23
    22 CALL SYMBOL(0.96,0.25,0.14,TITLE4,0.0,8)
       GO TO 24
    26 YLEN=9.0
       GO TO 29
   28  XLEN=9.0
       GO TO 25
    32 CALL ANN (DY,DY1,KEXP)
       EXPY=KEXP
       CALL SYMBOL(1.44,0.0,0.14,TITLE6,0.0,4)
       CALL NUMBER(0.96,0.0,0.14,DY1,0.0,2)
       CALL NUMBER(1.56,0.0,0.14,EXPY,0.0,-1)
       GO TO 23
    33 CALL ANN (DX,DX1,KEXP)
       EXPX=KEXP
       CALL SYMBOL(1.44,0.25,0.14,TITLE6,0.0,4)
       CALL NUMBER (0.96,0.25,0.14,DX1,0.0,2)
       CALL NUMBER (1.56,0.25,0.14,EXPX,0.0,-1)
       GO TO 24
       END

//LINK.SYSLMOD DD DSNAME=F0327.DSLPLOT(PLOTHALL),
// UNIT=2321,VOLUME=SER=CEL001,LABEL=EXPDT=74360,
// SPACE=(TRK,(90,5,1),RLSE),DISP=(NEW,KEEP)
```

```
// EXEC DSL4,REGION=150K
//DSL.INPUT DD *

CONTRL FINTIM=10E03,DELT=0.5,DELS=0.5
PRINT FREQ2,LFREQ,MAGH,ANGH,FREQ0,FREQ1
INTEG RKSFX
CONST NPLOT=3
INTGER NPLOT,NUM
RENAME TIME=FREQ
PARAM PI=3.14159
PARAM ZETA=0.01
INITIAL
      FREQ1=56.0
      DELS=0.5
      NUM=0
      FREQ0=2.0*PI*FREQ1
      BETA=3000.0
DERIVATIVE
      FREQ2=FREQ/(2.0*PI)
      A=FREQ0**2-FREQ**2
      B=FREQ0**2*BETA-(2.0*FREQ0*ZETA+BETA)*FREQ**2
      C=(FREQ0**2+2.0*FREQ0*ZETA*BETA)*FREQ-FREQ**3
      DE=SQRT(B**2+C**2)
      MAGH=ABS(A/DE)
DYNAMIC
      A1=A*B
      B1=-A*C
      ANGH=ATAN2(B1,A1)*180.0/PI
      IF(FREQ.GE.1.0) LFREQ=ALOG10(FREQ)
SAMPLE
      IF(FREQ.GE.500.0) DELS=10.0
      IF(FREQ.GE.1000.0) DELS=100.0
      IF(FREQ.LT.1.0) RETURN
      NUM=NUM+1
      CALL DRWG(1,1,NUM,LFREQ,ANGH)
      CALL DRWG(2,1,NUM,LFREQ,MAGH)
      CALL PRINT
TERMINAL
      CALL ENDRW(NPLOT)
      GO TO (1,2),NPLOT
    2 ZETA=0.1
```

```
      GO TO 3
    1 ZETA=0.2
    3 CALL RERUN
END
STOP

//G.FT04F001 DD SPACE=(CYL,(5,1)),DCB=BLKSIZE=1998
//PLOT.STEPLIB DD DSN=S1269.DSLPLOT
//PLOT.FT12F001  DD UNIT=SYSDA,SPACE=(CYL,(3,1)),
//   DCB=(RECFM=FBA,LRECL=28,BLKSIZE=1960)
//PLOT.SYSIN DD *
NOTE: TITLES APPEAR AFTER PLOT.SYSIN CARD
```

**** RANDOM NOISE GENERATOR ****

NOTE: THE RANDOM NUMBER GENERATOR USED IN THIS PROGRAM WAS SUPPLIED BY
MR.GERARD P. LEARMONTH OF THE NAVAL POSTGRADUATE SCHOOL COMPUTER FACILITY.

```
// EXEC FORTCALG,REGION.GO=200K
//FORT.SYSIN DD *

      DIMENSION TRASH1(10000),YAXIS(200)
      REAL*8 TRASH2,TRASH3,DELT8,HTR8,TRASH8,TRASH9,ITITLE(12)
      REAL LABEL
      DATA DELT/0.0001/,TRASH3/0.0/,DELT8/0.0001/,IX/30844/
      DATA LABEL/'JUNK'/,I1/1/,IFLAG/0/
      CALL OVFLOW
      READ(3,100,END=10)TRASH,I1
   11 NOPT=1000
      NOPT1=NOPT-100
    9 IF(IFLAG.EQ.0) GO TO 6
      CALL SNORM(IX,TRASH1(101),NOPT1)
      GO TO 7
    6 CALL SNORM(IX,TRASH1,NOPT)
    7 TRASH9=TRASH1(1)
      TRASH3=0.0
      DO 1 I=1,NOPT
      J1=I-1
      T=DELT*(I-1)
      CALL FILTER(T,HTR)
      TRASH8=TRASH1(I)
      HTR8=HTR
      TRASH2=DELT8/2.0*(TRASH9*HTR8+TRASH8*1.0)
      IF(J1.LT.2) GO TO 2
      J3=J1
      DO 3 K=2,J1
      TRASH8=TRASH1(J3)
      T1=DELT*(K-1)
      CALL FILTER(T1,HTR)
      HTR8=HTR
      TRASH3=TRASH3+HTR*TRASH8
    3 J3=J3-1
      TRASH3=DELT8*TRASH3
    2 TRASH=TRASH2+TRASH3
      TRASH=100.0*TRASH
```

```
      WRITE(3,100) TRASH,I1
1     I1=I1+1
100   FORMAT(1X,E14.8,I6)
      I2=NOPT-99
      IFLAG=1
      DO 5 I=1,100
      TRASH1(I)=TRASH1(I2)
5     I2=I2+1
      IF(I1.GT.20000) GO TO 8
      GO TO 9
 8    REWIND 3
      DO 4 I=1,200
      READ(3,100) TRASH1(I),NUM
      TRASH1(I)=-TRASH1(I)
4     YAXIS(I)=NUM
      YLEN=20.0
      MODE=4
      WRITE(6,102) MODE
102   FORMAT('0','MODE=',I2)
      ICOUNT=1
      DY=10.0
      NCUR=0
      YMIN=0.0
      READ(5,101)(ITITLE(I),I=1,6)
      READ(5,101)(ITITLE(I),I=7,12)
101   FORMAT(6A8)
      CALL DRWG(200,TRASH1,YAXIS,NCUR,LABEL,ITITLE,XMIN,DX,YMIN,DY,MODE,
     1YLEN,ICOUNT)
      WRITE(6,102) MODE
      STOP
 10   I1=I1+1
      WRITE(6,103) I1
103   FORMAT('0','CONTINUING WITH POINT NR.',I6)
      GO TO 11
      END


      SUBROUTINE FILTER(T,FOUT)
      REAL*8 T8,FOUT1,T1
      IF(T.EQ.0) GO TO 1
      T8=T
      T1=400.0*T8
      FOUT1=DSIN(T1)/T8*3.14159265
      FOUT=FOUT1
      RETURN
1     FOUT=1.0
      RETURN
      END
```

BIBLIOGRAPHY

1. J. L. Farrell, Simulation of a Minimum Variance Orbital Navigation System, Proc AIAA/ION Third Manned Space Flight Meeting, 1964.

2. Reference deleted.

3. General Dynamics, Pomona Division, Harmonic Coupling Between Control Loops and Elastic Loops in a Bi-Stable Autopilot, TM-337-23.6-8.

4. R. Kalman, On the General Theory of Control Systems, Proc. 1st Int. Cong. on Automatic Contr., London, Butterworth Scientific Publications, Vol. 1, 1961, p. 481.

5. C. J. McArdle, Adaptive Autopilot, paper presented at California State College at Los Angeles, April 1968.

6. W. M. Syn, N. N. Turner, D. G. Wyman, Digital Simulation Language (DSL) User Manual, IBM System/360, IBM, November 1968.

7. G. J. Thaler and M. P. Pastel, Analysis and Design of Nonlinear Feedback Control Systems, McGraw-Hill, 1962.

8. J. T. Tou, Modern Control Theory, New York, McGraw-Hill, 1964.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
2.	Professor G. J. Thaler Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	5
3.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
4.	Dr. K. Deckert IBM Monterey and Cottle Rds. San Jose, California	1
5.	Mr. Martin Dost IBM Monterey and Cottle Rds. San Jose, California	1
6.	Mr. Wei-Mun Syn IBM Monterey and Cottle Rds. San Jose, California	1
7.	Mr. Robert Hall General Dynamics Corp. Pomona Division Pomona, California	1
8.	LCDR D. G. MacDougall Naval Plant Representative's Office General Dynamics Corp. Pomona Division Pomona, California	1
9.	LT David P. Hall 239 Calle de La Selva Navato (Loma Verde), California	1
10.	Capt. R. W. Anderson (ret.) 24303 El Rico Place Diamond Bar, California 91765	1

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Simulation Study of the Roll Control Loop of an Operational Surface to Air Missile		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis; December 1973 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) David Parkin Hall		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		12. REPORT DATE December 1973 13. NUMBER OF PAGES 127
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Naval Postgraduate School Monterey, California 93940		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Digital Simulations	Dither
Adaptive Control	Roll Wander
Missile Roll Control Loop	DSL
Digitally Simulated White Noise	DSLPLOT

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This thesis studies the roll loop of an operational surface to air missile as an example of a typical self adaptive control system. The missile studied also exhibits a phenomenon called ROLL WANDER, a low frequency oscillation in missile roll orientation. Because of this, the thesis is developed in three major areas of study. First, is the study of the missile roll control loop. Second, is a study of the phenomenon in the

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601 |
(Page 1)

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(20.)

computer simulation. Third and finally is the implementation, in the computer simulation, of a self adaptive notch filter to suppress the occurrence of ROLL WANDER.

Thesis
H1474 Hall
c.1
17143
Simulation study of the roll control loop of an operational surface to air missile.